Cosmic Particle Telescope Data Acquisition System

Jack Hirschman
George Washington University, ECE6213
December 21, 2018

Table of Contents

Introduction 4

Theory/Algorithm 4

Architecture 6

Overview and Blocks 6

Implementation and Verification 8

State Diagrams 9

Controllers and Tasks 12

HDL Code 13

Overview 13

DAQ Top Module 14

DAQ Module 16

BCD to 7 Seg for 3 displays Module 24

BCD to 7 Seg for 2 displays Module 26

Serial Output Module 27

Synthesis 31

FPGA Path 31

ASIC Path 33

Testing 35

Functional Testing 35

Testbench Code 37

On Board Testing 40

Arduino Test Code Version 1 42

Arduino Test Code Version 2 44

Problems and Debugging 46

Conclusions 46

Appendix A 47

TCL Script 47

Constraints File 61

Appendix B 68

Overview 68

Timing Report 68

Area Report 93
Power Report 96
Appendix C 99
Appendix D 150

Introduction

This project develops the first version of the data acquisition (DAQ) system for the cosmic particle telescope being developed by GW astrophysics under the direction of Professor Sylvain Guiriec. The project's goal is to build a semi-portable detector setup consisting of three detectors along with the capabilities to count cosmic particle events incident on the detectors and the capabilities to extract information from these events, such as pulse shape. My role in the project is to develop the frontend electronics on the project for readout from the detectors and to develop the logic and processing modules needed to extract information. The analog-to-digital converters (ADC) and the analog circuitry that feeds them, though part of the overall project and effort, are not a part of this ECE6213 project. The project for ECE6213 is thus focused around the logic and pre-processing elements in the DAQ, all implemented in an FPGA in the real experiment but for the purposes of this class implemented both in FPGA and ASIC paths. For a fuller discussion of the complete project overview and additional design constraints needed for the cosmic particle telescope, see Appendix D.

The design process involved defining requirements, obtaining block level design, generating needed state machines, developing pre-synthesis code version 1 and simulation, synthesizing code version 1 for ASIC path, implementing code version 1 in FPGA, adapting and updating code version 1 based on hardware interface, developing pre-synthesis code version 2, and repeating those steps until final version.

Not too many major issues were encountered along the way, but a few are presented in the table below.

Major issues	**Solution**
Generating latched trigger	Handshake between start and stop modules
BCD counter on counts and dead_counts	Small typo fixed

Theory/Algorithm

The algorithm behind the trigger system is straightforward. When all the detectors are hit by a particle, the system will count this as a trigger assuming enough energy has been deposited. The trick is to ensure that it is the same particle striking all detectors. The particles of most interest in this experiment are muons.[1] Three detectors will be stacked on top of each other to form a cone of incidence. The detectors will be close enough together such that the particle will be incident on all detectors nearly simultaneously, at least at the level of clock speed for this project (50 Mhz implies 20 ns per sample period). Coincidence logic like this will also remove background noise as the probability of noise being identical on all detectors is low and should not be larger than a discrimination threshold set at the frontend analog electronics.

$$Trigger = Detector\ 1\ \&\ Detector\ 2\ \&\ Detector\ 3$$

Latched $Trigger = Detector\ 1\ \&\ Detector\ 2\ \&\ Detector\ 3$ & (not *BUSY*)

[1] Muons are of interest because this project is for outreach to local high schools. We want to demonstrate the concepts of particle detection and of relativity. Muons from cosmic showers are relativistic, and their detection on the surface of Earth can demonstrate time dilation and length contraction because, without relativity taken into consideration, muons would decay before they reached the surface.

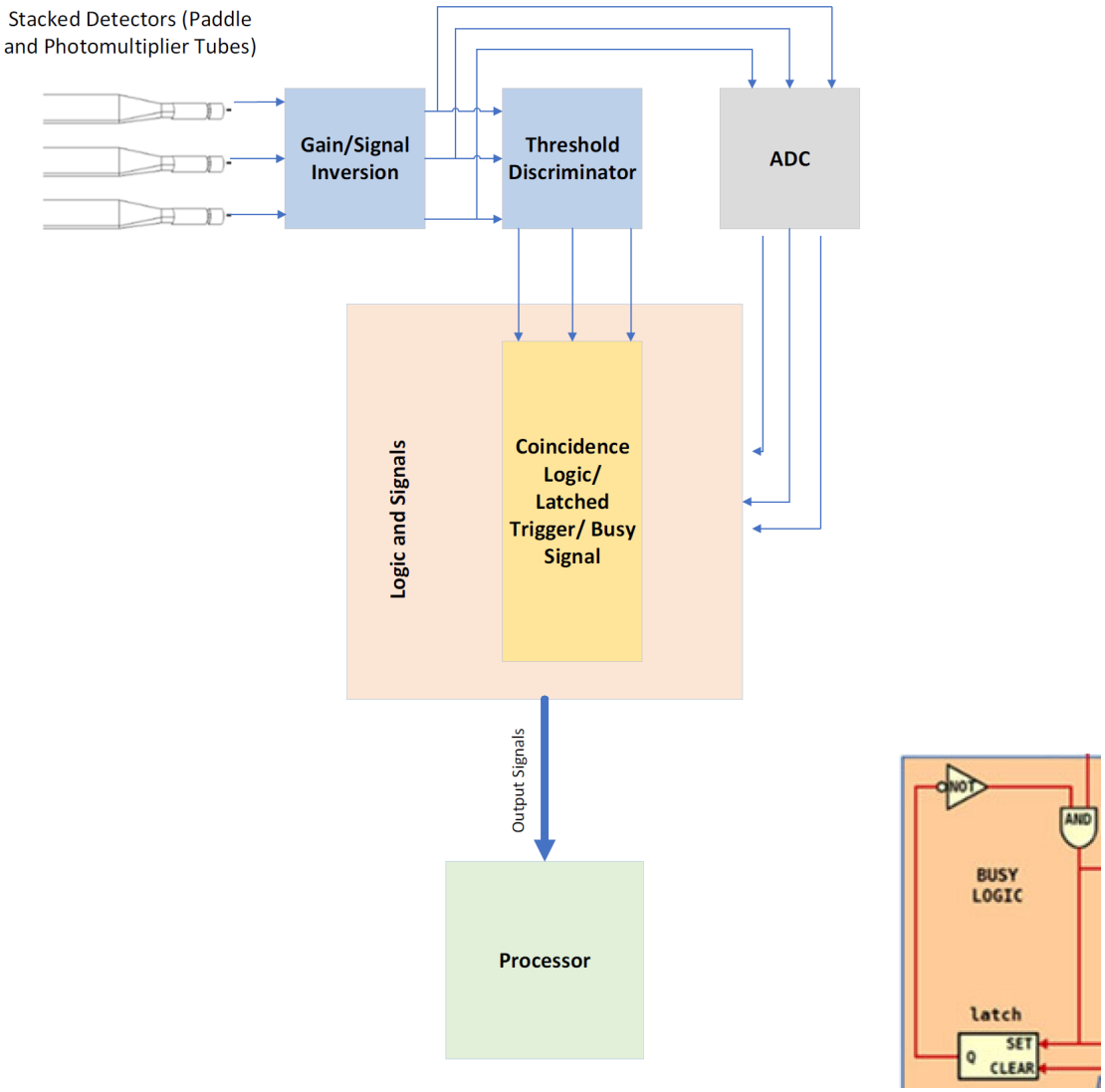

Figure 2.1: a) 3-channel overview of trigger detector setup and logic; b) More detailed view of coincidence and busy logic.

The trigger is used to start collection from an ADC. In this project's current design, this means that output from an ADC will feed into the FPGA, and the data will be continuously acquired and stored in buffers. A latched trigger will be derived from the initial trigger so that all of the ADC data corresponding to a particular event can be stored properly with time coherence between different paddles. The system will thus take time to work through all of the data provided by the ADC. The time required for this drives how long the latched trigger is active. While the latched trigger is active, a busy signal is generated to indicate that no more signals from the detectors will be taken as input. A diagram overview of the setup is shown in Figure 2.1. The figure on the left (a) is the whole system with detectors (consisting of scintillator paddles and photomultiplier tubes)

connecting to frontend analog electronics[2] which go to both the ADCs and the logic units.[3] Figure 2.1 (b) [4] presents a detailed view of the latched trigger and busy signal generation.

Architecture

Overview and Blocks

The project's design had to be simplified due to time constraints. Mainly, the ADC portion was not implemented.[5] Figure 3.1 shows an overview of the complete architecture, including external electronics and FPGA. Figure 3.2 shows the end-result block diagram of device under test for this report. Figure 3.3 presents a table of inputs and outputs and their associated descriptions.

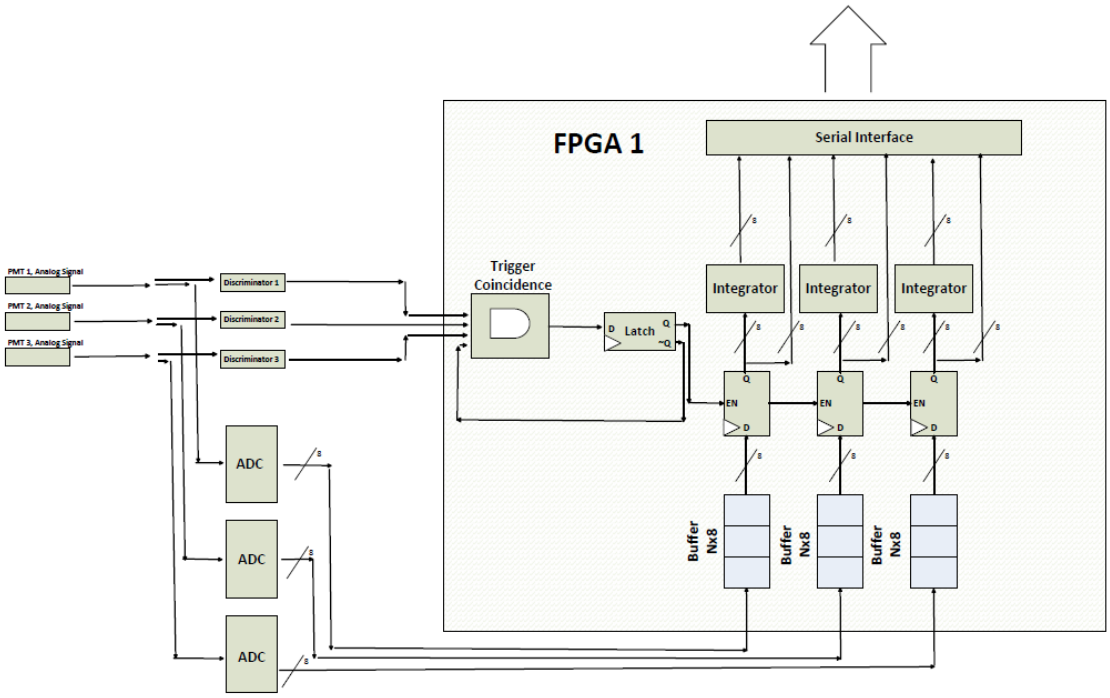

Figure 3.1: System architecture overview.

[2] With fast enough ADCs, discrimination may actually be done digitally; for the purposes of this report, it will be assumed frontend amplification and discrimination are carried-out with analog electronics.
[3] The signals from the amplification and inversion unit actually feed into a discriminator unit first, to qualify or dismiss signals based on a minimum voltage threshold. These digitized signals then go to the logic unit.
[4] Andrea Negri, "Introduction to Data AcQuisition," *ISOTDAQ 2018: 9th International School of Trigger and Data Acquisition*, https://indico.cern.ch/event/643308/contributions/2610520/attachments /1467722/2536718/isotdaq18.Negri.DaqIntro.pdf.
[5] The ADC portion was also excluded as it will likely be implemented using control and evaluation boards due to the complex nature of high speed ADCs (>500Msps, the sample rate likely needed for integration of pulse widths in the 10ns range). More on this will be dicussed in Appendix D.

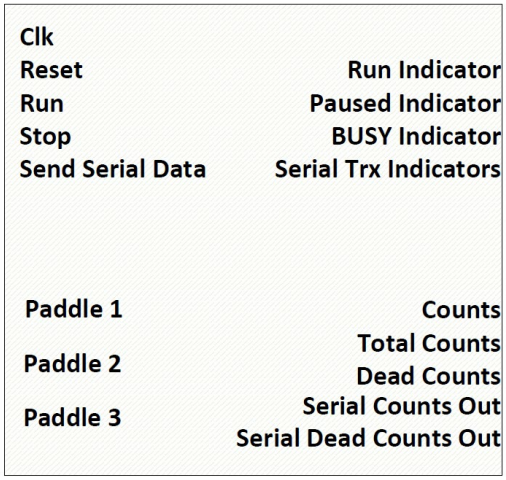

Figure 3.2: Block diagram of device under test (the FPGA or ASIC).

Pin Name	Input/Output	# of bits	Description
paddle_in_n	Input	3	Input from detectors (paddles)
startCount_n	Input	1	Starts counting triggers
stopCount_n	Input	1	Stops counting triggers
rst_in_n	Input	1	System reset
clk	Input	1	Clock input (50MHz)
sendSerialData_n	Input	1	Initiates serial data transmission
serialCount_trx_busy	Output	1	Indicates serial transmission of “counts” is occurring
serialDeadCount_trx_busy	Output	1	Indicates serial transmission of “dead counts” is occurring
runStateIndicator	Output	1	Indicates system is running
pausedStateIndicator	Output	1	Indicates system is paused/stopped
display_counts_out	Output	21	7-seg display of “counts”
display_deadCounts_out	Output	14	7-seg display of “dead counts”
display_totalCounts_out	Output	21	7-seg display of “total counts”
serial_count_out	Output	1	Transmission bit for serial transmission of “counts”
serial_deadCount_out	Output	1	Transmission bit for serial transmission of “deadCounts”
counts	Output	10	Binary count of “counts”
deadCounts	Output	10	Binary count of “dead counts”
totalCounts	Output	10	Binary count of “total counts”
busy	Output	1	Trigger actively latched indicator/system working with an event

Figure 3.3: Table of inputs and outputs for design.

Implementation and Verification

There were three stages of development, with verification at each stage. The first stage involved developing code and testing using a testbench and simulations. These were pre-synthesis simulations aimed at general functionality of the design. All simulations were carried-out using Cadence.

The next stage of testing involved simulation of the synthesized design, which was aimed at the ASIC path. Here, Synopsys was used for logic synthesis, optimization, place and route, and static timing analysis. See Figure 3.4 for the design flow diagram of this stage.[6]

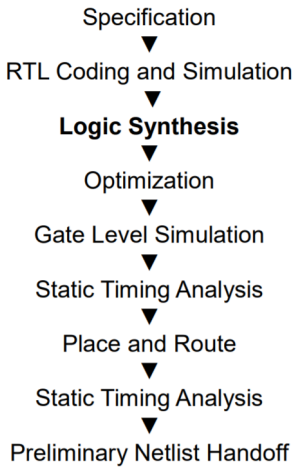

Figure 3.4: Design flow diagram.

The third stage involved creating a project in Quartus Prime for implementing the design in an Altera FPGA, DE2-115 development board. Place and route as well as static timing were completed again at this stage as the device constraint file was added. This file routed inputs and outputs in the design to peripherals on the development board. Simulations were not repeated at this stage. A programming bit stream was generated and used to load the design onto the FPGA. Initial tests were carried out using push-buttons to simulate detector inputs to the design. For the final tests, an Arduino was used to feed-in test inputs. An oscilloscope was used to check serial output functionality of the design as well.

State Diagrams

The two main state machines in the design are related to user interface. The most important one is the state machine for moving among the start counting state, the stop counting state, and the clear state. In hardware, each state is entered via a push-button, so in the code the design looks for changes in the input signals.[7] This state diagram is shown in Figure 3.5.

The second state machine is for serial transmission. It has two states, sending data and not sending data. The system enters the sending state via a push-button switch. The system then goes through the sending protocol and automatically exits the sending state once transmission has completed. The state diagram for this state machine is shown in Figure 3.6. An algorithmic state machine for the serial transmission protocol is shown in Figure 3.7. The control state machine initiates the transmission protocol.

[6] Jerry Wu, “ECE 6213 – Synopsys Tutorial: Using the Design Compiler,” ECE6213, 2018.
[7] The implementation looks for falling edges as the push-button switches use active low signaling.

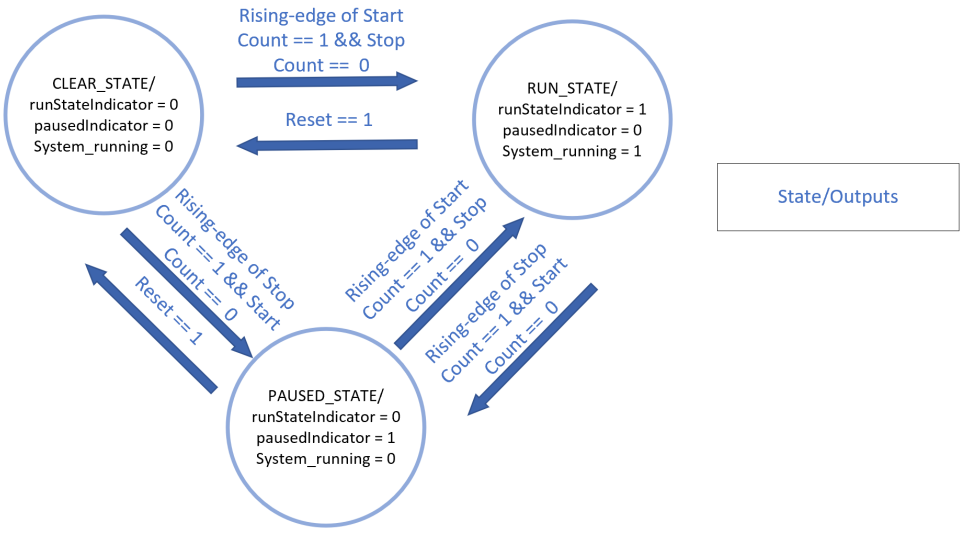

Figure 3.5: State machine for starting, stopping, and clearing (resetting) coincidence count.

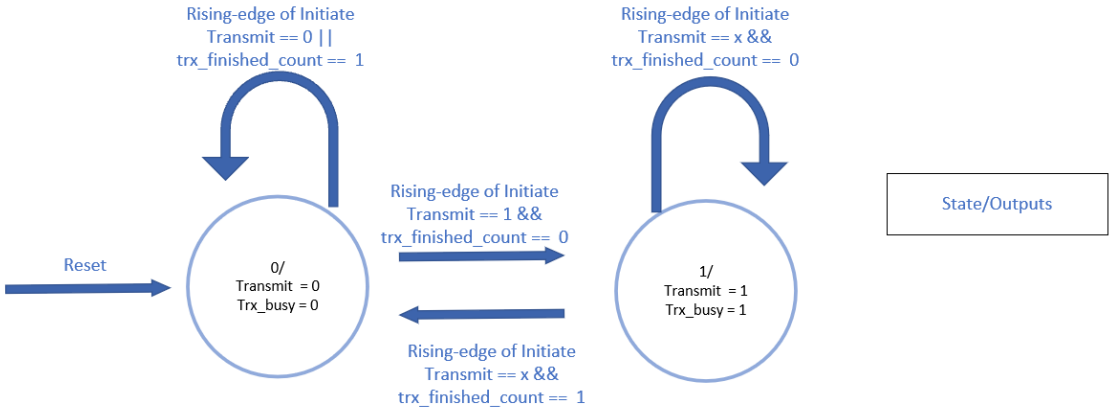

Figure 3.6: State machine for starting, stopping, and resetting the serial transmission protocol.

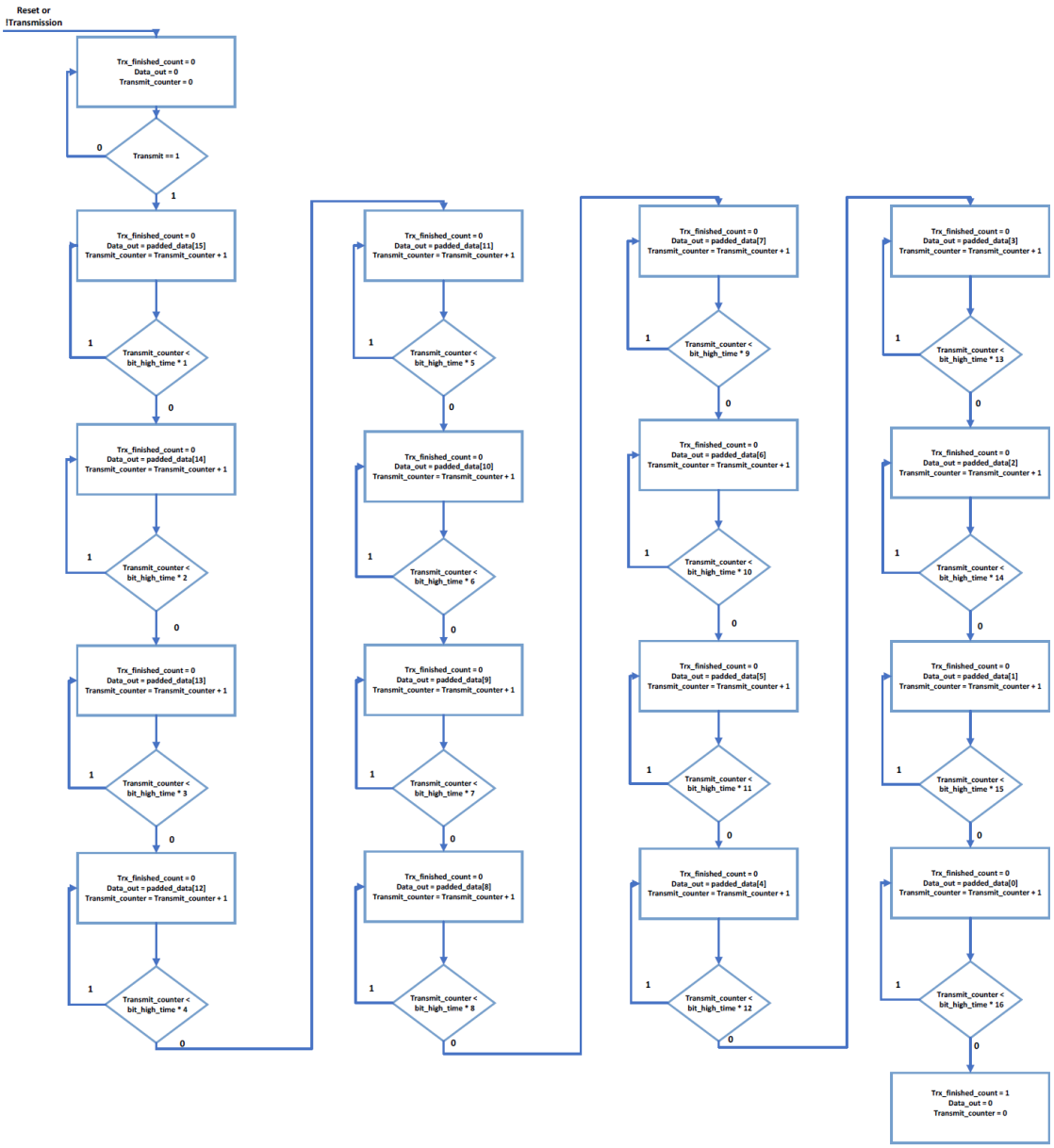

Figure 3.7: Algorithmic flow diagram for serial transmission. Note, if the transmit signal from control state machine (Figure 3.6) goes low and inactive, the transmission protocol is stopped.

The only other state machine of note is the latched trigger state machine. This one works similarly to the serial transmission control where the latched trigger is initiated and then stays active until a counter reaches its terminal count.[8] This is exactly like a simplified version of Figure 3.7 where

[8] The terminal count value is chosen in the design. The latched trigger should stay active for the amount of time needed to capture the pulse shape of the detector signals using the ADC's. Since the ADC delectronics have not be completed yet, the time the latched trigger remains active in this project is arbitrary and serves as a proof-of-concept.

the system only looks for a finishing count instead of incrementing steps. Therefore, this diagram for the latched trigger will not be presented here.

Controllers and Tasks

Task	Description
Control State Machine for User Interface	Reset, start, and stop for system counting as well as LED indication of current state (green LED means running, red LED means stopped, no LED on means reset).
Coincidence Trigger Generation	Combinational AND of three detector input signals
Latched Trigger Generation	Same as trigger but stays high for set amount of time (this corresponds to a BUSY signal).
Counts Incrementation	Counts output increments at rising-edge of each latched trigger.
Dead Counts Incrementation	Dead counts output increments at rising-edge of each trigger while the latched trigger is high.
Total Counts Incrementation	Total counts output increments at rising-edge of every trigger (is sum of counts and dead counts).
Serial Data Output Controller	Controller takes user input to begin serial transmission protocol. Indicates to user (via green LEDs) that transmission of both counts and dead counts in progress.
Serial Data Output	Saves current value of counts and dead counts in registers and pads these values with 110xxxxxxxxx001 where the x's represent the actual value. The protocol then transmits from MSB to LSB one bit at a time, keep each bit at its value for roughly 1 second (this can be adjusted).
BCD to 7-Seg Display	Converts binary coded decimal values (which are generated in counts, dead counts, and total counts routines) into 7-seg display outputs.

Figure 3.8: Table of completed tasks in project and associated descriptions.

The main controllers and tasks in the design are presented in tabular form in Figure 3.8. Each has an associated description. More details about several of the controllers are given in the previous section on State Diagrams.

HDL Code

Overview

The entire code set includes two files. The first file is the top module which samples and synchronizes input signals and instantiates the main module in the hierarchy. The remaining code, which instantiates the main module, is made of four modules all included in the second file. All code was written in Verilog. The two files contain only synthesizable code but are themselves not synthesized at this stage. Each module from the synthesizable code will be listed on a separate page. A testbench of code instantiates the top module of the design and will be discussed in Section 6.2.

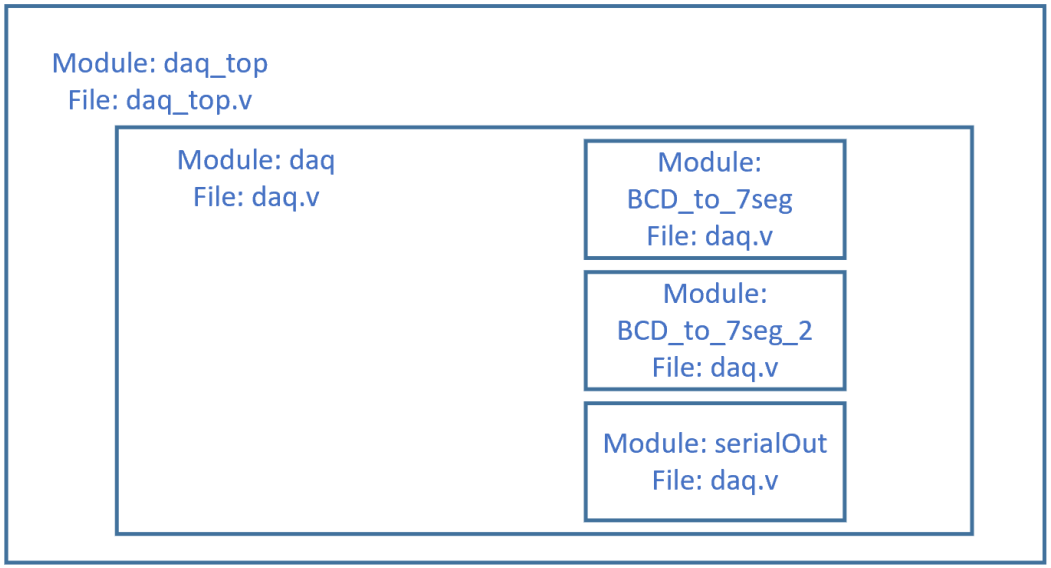

Figure 4.1: Graphic of synthesizable modules and file structure.

DAQ Top Module

Verilog Module Name: daq_top.v

Implementation of module: Datapath, shown in figure 4.1

```
`timescale 1ns/1ps
module daq_top(paddle_in_n, startCount_n, stopCount_n, rst_in_n, clk, sendSerialData_n,
      serialCount_trx_busy, serialDeadCount_trx_busy, runStateIndicator, pausedIndicator,
      display_totalCounts_out, display_counts_out, display_deadCounts_out, serial_count_out,
      serial_deadCount_out, counts, deadCounts, totalCounts, busy);
      /****inputs******/
      input             rst_in_n, clk, startCount_n, stopCount_n, sendSerialData_n; //active low reset,
                        clk, count initiate, count stop
      input [2:0]       paddle_in_n; //input from three sepatate paddles: paddle_0, paddle_1, paddle_2,
                        active low

      /*****outputs******/
      output            runStateIndicator, pausedIndicator, busy;
      output [20:0]     display_totalCounts_out, display_counts_out; //output to 3 seven-seg displays
      output [13:0]     display_deadCounts_out; //output to two 7-segs
      output            serial_count_out, serial_deadCount_out; //serial transmission output bit
      output            serialCount_trx_busy, serialDeadCount_trx_busy; //serial transmission indicators
      output [9:0]      counts, deadCounts, totalCounts; //binary counters


      /**** signal types declarations *****/
      wire              runStateIndicator, pausedIndicator;
      wire              serialCount_trx_busy, serialDeadCount_trx_busy, busy;
      wire              serial_count_out, serial_deadCount_out;
      wire [9:0]        counts, deadCounts, totalCounts;
      wire              rst; //reset signal to change to active high

      reg [2:0]         reset_gen; //derived signal from rst_in_n
      reg [2:0]         paddle_in_syncd_n; //synced detector signals
      reg               startCount_gen_n, stopCount_gen_n; //derived start and stop count signals

      /****** Instantiating daq module *******/
      Daq U1(.paddle_in_n(paddle_in_syncd_n), .startCount_n(startCount_gen_n), .stopCount_n
            (stopCount_gen_n), .rst(reset_gen[2]), .clk(clk), .sendSerialData_n
```

```
            (sendSerialData_n), .serialCount_trx_busy(serialCount_trx_busy),
            .serialDeadCount_trx_busy(serialDeadCount_trx_busy), .runStateIndicator
            (runStateIndicator), .pausedIndicator(pausedIndicator), .display_totalCounts_out
            (display_totalCounts_out), .display_counts_out(display_counts_out),
            .display_deadCounts_out(display_deadCounts_out), .serial_count_out
            (serial_count_out), .serial_deadCount_out(serial_deadCount_out),
            .counts(counts), .deadCounts(deadCounts), .totalCounts(totalCounts), .busy(busy));
      //reset clock in
      assign rst = ~rst_in_n; //change to posedge
      always @(posedge clk or posedge rst) begin
            if (rst == 1)
                  reset_gen <= 3'b111; //creating more stable reset signal
            else
                  reset_gen <= reset_gen << 1; //reset must be low for three clock periods (stable)
      end
      //clock in signals
      //uses MSB of reset_gen for stability purposes
      always @(posedge clk or posedge reset_gen[2]) begin
            if (reset_gen[2] == 1) begin
                  paddle_in_syncd_n <= 3'b111;
                  startCount_gen_n <= 1'b1;
                  stopCount_gen_n <= 1'b1;
            end
            else begin
                  paddle_in_syncd_n <= paddle_in_n;
                  startCount_gen_n <= startCount_n;
                  stopCount_gen_n <= stopCount_n;
            end
      end
endmodule
```

DAQ Module

Verilog Module Name: daq.v

Implementation of module: Datapath, shown in figure 4.1

```
`timescale 1ns/1ps
module daq(paddle_in_n, startCount_n, stopCount_n, rst, clk, sendSerialData_n, serialCount_trx_busy,
      serialDeadCount_trx_busy, runStateIndicator, pausedIndicator, display_totalCounts_out,
      display_counts_out, display_deadCounts_out, serial_count_out, serial_deadCount_out, counts,
      deadCounts, totalCounts, busy);
      /*** Parameters  ****/
      parameter LATCHED_TIME = 100000000; //Gate time for latch on latched trigger; essentially how long
            the system will be busy for; chosen to be noticeable in hardware implementation
      parameter LATCHED_TIME_BITS = 30; //**** Number of bits in counter needed for LATCHED_TIME
      parameter CLEAR_STATE = 2'b00, RUN_STATE = 2'b01, PAUSED_STATE = 2'b10; //States in user controller

      /*** Inputs   *****/
      input             rst, clk, startCount_n, stopCount_n, sendSerialData_n; //active low reset, clk,
                        count initiate, count stop
      input [2:0]       paddle_in_n; //input from three sepatate paddles: paddle_0, paddle_1, paddle_2,
                        active low
      /**** Outputs *****/
      output            runStateIndicator, pausedIndicator, serialCount_trx_busy, serialDeadCount_trx_busy,
                        busy;
      output [20:0]     display_totalCounts_out, display_counts_out; //output to 3 seven-seg displays
      output [13:0]     display_deadCounts_out; //output to 2 seven seg displays
      output            serial_count_out, serial_deadCount_out;
      output [9:0]      totalCounts, counts, deadCounts;

      /**** Sginal type declarations ****/
      reg               runStateIndicator, pausedIndicator; //output for LED indicators
      wire              serialCount_trx_busy, serialDeadCount_trx_busy;
      wire [20:0]       display_totalCounts_out, display_counts_out;
      wire [13:0]       display_deadCounts_out;
      wire [20:0]       display_totalCounts, display_counts;
      wire [13:0]       display_deadCounts;
      wire              serial_count_out, serial_deadCount_out;

      reg               trigger, trigger_dly, trigger_latched, trigger_latched_dly, busy;
```

```
reg [2:0]         paddle_in_n_dly, paddle_in_trigger;
reg               startCount_n_dly, stopCount_n_dly;

reg [1:0]         systemState, systemState_next;
reg               system_running;
reg               start_latch_counter;
reg               finishedCount;
reg [9:0]         counts, deadCounts, totalCounts; //counts: number of particles while trigger
                  activated and busy low, deadCounts: number of triggers that occur while busy
                  enabled
reg [11:0]        totalCounts_bcd, counts_bcd; //for displaying three numbers
reg [7:0]         deadCounts_bcd; //displaying two digits
reg [LATCHED_TIME_BITS - 1:0] latch_counter; //determines how long the system is busy
reg               sendSerialData_dly;
wire              sendSerialData;
reg[9:0]          count_captured, deadCount_captured; //used for sampling counts before serial trx

/*** Changing serial data send signal to active high ****/
assign sendSerialData = ~sendSerialData_n;

/**** Trigger Portion ******/
//If all inputs are low (i.e. three detectors received a signal), trigger goes high
always @(paddle_in_n) begin
      if (paddle_in_n == 3'b000)
            trigger <= 1;
      else
            trigger <= 0;
end
//Creates delayed trigger signal for edge detection
always @(posedge clk or posedge rst) begin
      if (rst == 1)
            trigger_dly <= 0;
      else
            trigger_dly <= trigger;
end

/********** Latched Trigger  ******/
//starts latch counter and controls latched trigger value
```

```
always @(posedge clk or posedge rst) begin
      if (rst == 1) begin
            start_latch_counter <= 0;
            trigger_latched <= 0;
            busy <= 0;
      end
      else if (trigger_dly == 0 && trigger == 1 && busy == 0 && system_running == 1) begin
            start_latch_counter <= 1;
            trigger_latched <= 1;
            busy <= 1;
      end
      else if (system_running == 1 && finishedCount == 0) begin
            start_latch_counter <= start_latch_counter;
            trigger_latched <= trigger_latched;
            busy <= busy;
      end
      else begin
            start_latch_counter <= 0;
            trigger_latched <= 0;
            busy <= 0;
      end
end
//stops latch counter to feed back into previous procedure
always @(posedge clk or posedge rst) begin
      if (rst == 1) begin
            latch_counter <= 0;
            finishedCount <= 0;
      end
      else if (start_latch_counter == 1)
            if (latch_counter >= (LATCHED_TIME - 1)) begin
                  latch_counter <= 0;
                  finishedCount <= 1;
            end
            else begin
                  latch_counter <= latch_counter + 1;
                  finishedCount <= 0;
            end
      else begin
```

```
            latch_counter <= 0;
            finishedCount <= 0;
      end
end
//creates delayed trigger latched for edge detection
always @(posedge clk or posedge rst) begin
      if (rst == 1)
            trigger_latched_dly <= 0;
      else
            trigger_latched_dly <= trigger_latched;
end

/******** System State **********/
//Run state, Paused State, Reset State

//next state transition and relevant delay signal assignment
always @(posedge clk or posedge rst) begin
      if (rst == 1) begin
            systemState <= CLEAR_STATE;
            startCount_n_dly <= 1;
            stopCount_n_dly <= 1;
      end
      else begin
            systemState <= systemState_next;
            startCount_n_dly <= startCount_n;
            stopCount_n_dly <= stopCount_n;
      end
end

//next state logic
always @(startCount_n or startCount_n_dly or stopCount_n or stopCount_n_dly or systemState) begin
      if (startCount_n == 0 && startCount_n_dly == 1 && stopCount_n == 1 && stopCount_n_dly == 1)
      //running state initiated by startCount button pressed and stop count button not on
            systemState_next <= RUN_STATE;
      else if (startCount_n == 1 && startCount_n_dly == 1 && stopCount_n == 0 && stopCount_n_dly ==
      1)
            systemState_next <= PAUSED_STATE;
      else
```

```
            systemState_next <= systemState;
end

//output and indicator logic
always @(systemState) begin
      if (systemState == CLEAR_STATE) begin
            runStateIndicator <= 0;
            pausedIndicator <= 0;
            system_running <= 0;
      end
      else if (systemState == RUN_STATE) begin
            runStateIndicator <= 1;
            pausedIndicator <= 0;
            system_running <= 1;
      end
      else if (systemState == PAUSED_STATE) begin
            runStateIndicator <= 0;
            pausedIndicator <= 1;
            system_running <= 0;
      end
      else begin
            runStateIndicator <= 0;
            pausedIndicator <= 0;
            system_running <= 0;
      end
end

/***************  Counting   ********/

//using bcd to seven seg module
BCD_to_7seg totalCountsConvert(.bcd(totalCounts_bcd), .segs(display_totalCounts_out));
BCD_to_7seg countsConvert(.bcd(counts_bcd), .segs(display_counts_out));
BCD_to_7seg_2 deadCountsConvert(.bcd(deadCounts_bcd), .segs(display_deadCounts_out));



always @(posedge clk or posedge rst) begin
      if (rst == 1) begin
```

```
		counts <= 0;
		deadCounts <= 0;
		totalCounts <= 0;
		counts_bcd <= 0;
		deadCounts_bcd <= 0;
		totalCounts_bcd <= 0;
	end
	else if (system_running) begin
		//total counts portion (every rising edge of trigger increments total counts)
		if (trigger == 1 && trigger_dly == 0) begin
			totalCounts <= totalCounts + 1; //binary count
			if (totalCounts_bcd[3:0] == 4'd9) begin //bcd count
				totalCounts_bcd[3:0] <= 0;
				if (totalCounts_bcd[7:4] == 4'd9) begin
					totalCounts_bcd[7:4] <= 0;
					if (totalCounts_bcd[11:8] == 4'd9)
						totalCounts_bcd[11:8] <= 0;
					else
						totalCounts_bcd[11:8] <= totalCounts_bcd[11:8] + 1;
				end
				else
					totalCounts_bcd[7:4] <= totalCounts_bcd[7:4] + 1;
			end
			else
				totalCounts_bcd[3:0] <= totalCounts_bcd[3:0] + 1;
		end
		else begin
			totalCounts <= totalCounts;
			totalCounts_bcd <= totalCounts_bcd;
		end
		//latched trigger portion (every rising edge latched trigger increments counts)
		if (trigger_latched == 1 && trigger_latched_dly == 0) begin
			counts <= counts + 1;
			if (counts_bcd[3:0] == 4'd9) begin //bcd count
				counts_bcd[3:0] <= 0;
				if (counts_bcd[7:4] == 4'd9) begin
					counts_bcd[7:4] <= 0;
					if (counts_bcd[11:8] == 4'd9)
```

```
                              counts_bcd[11:8] <= 0;
                        else
                              counts_bcd[11:8] <= counts_bcd[11:8] + 1;
                  end
                  else
                        counts_bcd[7:4] <= counts_bcd[7:4] + 1;
            end
            else
                  counts_bcd[3:0] <= counts_bcd[3:0] + 1;
      end
      else begin
            counts_bcd <= counts_bcd;
            counts <= counts;
      end
      //dead counts portion (every trigger while latched trigger is high increments dead
      counts)
      if (trigger_latched == 1 && trigger == 1 && trigger_dly == 0) begin
            deadCounts <= deadCounts + 1;
            if (deadCounts_bcd[3:0] == 9) begin //TEST
                  deadCounts_bcd[3:0] <= 0;
                  if (deadCounts_bcd[7:4] == 9) begin
                        deadCounts_bcd[7:4] <= 0;
                  end
                  else begin
                        deadCounts_bcd[7:4] <= deadCounts_bcd[7:4] + 1;
                  end
            end
            else
                  deadCounts_bcd[3:0] <= deadCounts_bcd[3:0] + 1;
      end
      else begin
            deadCounts <= deadCounts;
            deadCounts_bcd <= deadCounts_bcd;
      end
end
else begin //(systemState == PAUSED_STATE) begin
      totalCounts <= totalCounts;
      counts <= counts;
```

```
                deadCounts <= deadCounts;
                totalCounts_bcd <= totalCounts_bcd;
                counts_bcd <= counts_bcd;
                deadCounts_bcd <= deadCounts_bcd;
          end
     end

     /******* Data Transmit ************/
     serialOut serialCountOut(.clk(clk), .rst(rst), .initiate(sendSerialData), .data_in
          (count_captured), .data_out(serial_count_out), .trx_busy(serialCount_trx_busy));
     serialOut serialDeadCountOut(.clk(clk), .rst(rst), .initiate(sendSerialData), .data_in
          (deadCount_captured), .data_out(serial_deadCount_out), .trx_busy(serialDeadCount_trx_busy));

     always @(posedge clk or posedge rst) begin
          if (rst) begin
                count_captured <= 0;
                deadCount_captured <= 0;
          end
          else if (sendSerialData == 1 && sendSerialData_dly == 0 && serialCount_trx_busy == 0 &&
          serialDeadCount_trx_busy == 0) begin
                count_captured <= counts;
                deadCount_captured <= deadCounts;
          end
          else begin
                count_captured <= count_captured;
                deadCount_captured <= deadCount_captured;
          end
     end

     always @(posedge clk or posedge rst) begin
          if (rst)
                sendSerialData_dly <= 0;
          else
                sendSerialData_dly <= sendSerialData;
     end
endmodule
```

BCD to 7 Seg for 3 displays Module

Verilog Module Name: BCD_to_7seg module within daq.v

Implementation of module: Datapath, shown in figure 4.1

```
module BCD_to_7seg (bcd, segs);
    input [11:0]        bcd;
    output [20:0]       segs;
       reg [20:0]       segs;

    always @(bcd)  begin
       case (bcd[3:0])
                    0 : segs[6:0] = ~7'b0111111;
                    1 : segs[6:0] = ~7'b0000110;
                    2 : segs[6:0] = ~7'b1011011;
                    3 : segs[6:0] = ~7'b1001111;
                    4 : segs[6:0] = ~7'b1100110;
                    5 : segs[6:0] = ~7'b1101101;
                    6 : segs[6:0] = ~7'b1111101;
                    7 : segs[6:0] = ~7'b0000111;
                    8 : segs[6:0] = ~7'b1111111;
                    9 : segs[6:0] = ~7'b1100111;

                    default : segs[6:0] = ~7'b0000001;
       endcase
      case (bcd[7:4])
                    0 : segs[13:7] = ~7'b0111111;
                    1 : segs[13:7] = ~7'b0000110;
                    2 : segs[13:7] = ~7'b1011011;
                    3 : segs[13:7] = ~7'b1001111;
                    4 : segs[13:7] = ~7'b1100110;
                    5 : segs[13:7] = ~7'b1101101;
                    6 : segs[13:7] = ~7'b1111101;
                    7 : segs[13:7] = ~7'b0000111;
                    8 : segs[13:7] = ~7'b1111111;
                    9 : segs[13:7] = ~7'b1100111;

                    default : segs[13:7] = ~7'b0000001;
       endcase
```

```
            case (bcd[11:8])
                 0 : segs[20:14] = ~7'b0111111;
                 1 : segs[20:14] = ~7'b0000110;
                 2 : segs[20:14] = ~7'b1011011;
                 3 : segs[20:14] = ~7'b1001111;
                 4 : segs[20:14] = ~7'b1100110;
                 5 : segs[20:14] = ~7'b1101101;
                 6 : segs[20:14] = ~7'b1111101;
                 7 : segs[20:14] = ~7'b0000111;
                 8 : segs[20:14] = ~7'b1111111;
                 9 : segs[20:14] = ~7'b1100111;

                 default : segs[20:14] = ~7'b0000001;
            endcase
      end
endmodule
```

BCD to 7 Seg for 2 displays Module

Verilog Module Name: BCD_to_7seg_2 module within daq.v

Implementation of module: Datapath, shown in figure 4.1

```
module BCD_to_7seg_2 (bcd, segs);
    input [7:0]         bcd;
    output [13:0]       segs;
    reg [13:0]          segs;

    always @(bcd)  begin
       case (bcd[3:0])
                    0 : segs[6:0] = ~7'b0111111;
                    1 : segs[6:0] = ~7'b0000110;
                    2 : segs[6:0] = ~7'b1011011;
                    3 : segs[6:0] = ~7'b1001111;
                    4 : segs[6:0] = ~7'b1100110;
                    5 : segs[6:0] = ~7'b1101101;
                    6 : segs[6:0] = ~7'b1111101;
                    7 : segs[6:0] = ~7'b0000111;
                    8 : segs[6:0] = ~7'b1111111;
                    9 : segs[6:0] = ~7'b1100111;
                    default : segs[6:0] = ~7'b0000001;
      endcase
      case (bcd[7:4])
                    0 : segs[13:7] = ~7'b0111111;
                    1 : segs[13:7] = ~7'b0000110;
                    2 : segs[13:7] = ~7'b1011011;
                    3 : segs[13:7] = ~7'b1001111;
                    4 : segs[13:7] = ~7'b1100110;
                    5 : segs[13:7] = ~7'b1101101;
                    6 : segs[13:7] = ~7'b1111101;
                    7 : segs[13:7] = ~7'b0000111;
                    8 : segs[13:7] = ~7'b1111111;
                    9 : segs[13:7] = ~7'b1100111;
                    default : segs[13:7] = ~7'b0000001;
       endcase
      end
endmodule
```

Serial Output Module

Verilog Module Name: serialOut module within daq.v

Implementation of module: Datapath, shown in figure 4.1

```
module serialOut (clk, rst, initiate, data_in, data_out, trx_busy);
      input             clk, rst, initiate;
      input [9:0]       data_in;
      output            data_out, trx_busy;
      reg               data_out, trx_busy; //trx_busy is redundant with trasmit signal
                        but preferred clearer name for passing to outside module

      wire [15:0]       padded_data; //16 bit word whose MSB will be transmitted
      reg               initiate_dly;
      reg [29:0]        trx_delay_counter; total counter for transmission process
      reg               trx_finished_count, transmit;

      //time serial output high for each transmitted bit
      parameter SERIAL_PROTOCOL_TIME = 50000000; //1 sec on 50MHz clock; value each bit will be held for

      //prepare a data word
      assign padded_data = {3'b110, data_in, 3'b001}; //16 bits; giving a pattern of 110xxxxxxxx001
      //transmit controller state machine; uses handshake via trx_finished_count with next procedure
      always @(posedge clk or posedge rst) begin
            if (rst) begin
                  transmit <= 0;
                  trx_busy <= 0;
            end
            else if (initiate == 1 && initiate_dly == 0 && trx_finished_count == 0) begin
                  transmit <= 1;
                  trx_busy <= 1;
            end
            else if (trx_finished_count == 1) begin
                  transmit <= 0;
                  trx_busy <= 0;
            end
            else begin
                  transmit <= transmit;
                  trx_busy <= trx_busy;
```

```
      end
end
//counter and output bit selector
//transmission selects MSB of padded data, waits the selected delay, and then selects next.
//iterates through this process until 16 bit word is transmitted.
always @(posedge clk or posedge rst) begin
      if (rst) begin
            trx_finished_count <= 0;
            data_out <= 0;
            trx_delay_counter <= 0;
      end
      //enter transmit cycle and check value of counter; counter will increment until 30 bit reg
      full; every 50M counts change value of data_out
      else if (transmit == 1) begin
            if (trx_delay_counter < SERIAL_PROTOCOL_TIME*(1)) begin
                  trx_finished_count <= 0;
                  data_out <= padded_data[15];
                  trx_delay_counter <= trx_delay_counter + 1;
            end
            else if (trx_delay_counter < SERIAL_PROTOCOL_TIME*(2)) begin
                  trx_finished_count <= 0;
                  data_out <= padded_data[14];
                  trx_delay_counter <= trx_delay_counter + 1;
            end
            else if (trx_delay_counter < SERIAL_PROTOCOL_TIME*(3)) begin
                  trx_finished_count <= 0;
                  data_out <= padded_data[13];
                  trx_delay_counter <= trx_delay_counter + 1;
            end
            else if (trx_delay_counter < SERIAL_PROTOCOL_TIME*(4)) begin
                  trx_finished_count <= 0;
                  data_out <= padded_data[12];
                  trx_delay_counter <= trx_delay_counter + 1;
            end
            else if (trx_delay_counter < SERIAL_PROTOCOL_TIME*(5)) begin
                  trx_finished_count <= 0;
                  data_out <= padded_data[11];
                  trx_delay_counter <= trx_delay_counter + 1;
```

```
end
else if (trx_delay_counter < SERIAL_PROTOCOL_TIME*(6)) begin
      trx_finished_count <= 0;
      data_out <= padded_data[10];
      trx_delay_counter <= trx_delay_counter + 1;
end
else if (trx_delay_counter < SERIAL_PROTOCOL_TIME*(7)) begin
      trx_finished_count <= 0;
      data_out <= padded_data[9];
      trx_delay_counter <= trx_delay_counter + 1;
end
else if (trx_delay_counter < SERIAL_PROTOCOL_TIME*(8)) begin
      trx_finished_count <= 0;
      data_out <= padded_data[8];
      trx_delay_counter <= trx_delay_counter + 1;
end
else if (trx_delay_counter < SERIAL_PROTOCOL_TIME*(9)) begin
      trx_finished_count <= 0;
      data_out <= padded_data[7];
      trx_delay_counter <= trx_delay_counter + 1;
end
else if (trx_delay_counter < SERIAL_PROTOCOL_TIME*(10)) begin
      trx_finished_count <= 0;
      data_out <= padded_data[6];
      trx_delay_counter <= trx_delay_counter + 1;
end
else if (trx_delay_counter < SERIAL_PROTOCOL_TIME*(11)) begin
      trx_finished_count <= 0;
      data_out <= padded_data[5];
      trx_delay_counter <= trx_delay_counter + 1;
end
else if (trx_delay_counter < SERIAL_PROTOCOL_TIME*(12)) begin
      trx_finished_count <= 0;
      data_out <= padded_data[4];
      trx_delay_counter <= trx_delay_counter + 1;
end
else if (trx_delay_counter < SERIAL_PROTOCOL_TIME*(13)) begin
      trx_finished_count <= 0;
```

```
                        data_out <= padded_data[3];
                        trx_delay_counter <= trx_delay_counter + 1;
                  end
                  else if (trx_delay_counter < SERIAL_PROTOCOL_TIME*(14)) begin
                        trx_finished_count <= 0;
                        data_out <= padded_data[2];
                        trx_delay_counter <= trx_delay_counter + 1;
                  end
                  else if (trx_delay_counter < SERIAL_PROTOCOL_TIME*(15)) begin
                        trx_finished_count <= 0;
                        data_out <= padded_data[1];
                        trx_delay_counter <= trx_delay_counter + 1;
                  end
                  else if (trx_delay_counter < SERIAL_PROTOCOL_TIME*(16)) begin
                        trx_finished_count <= 0;
                        data_out <= padded_data[0];
                        trx_delay_counter <= trx_delay_counter + 1;
                  end
                  else begin // (trx_delay_counter >= SERIAL_PROTOCOL_TIME*16)
                        trx_finished_count <= 1;
                        data_out <= 0;
                        trx_delay_counter <= 0;
                  end
            end
            else begin
                  trx_finished_count <= 0; //clear finished count so can transmit again
                  data_out <= 0; //output zero if no data
                  trx_delay_counter <= 0; //clear counter for next transmit
            end
      end
      //generate trigger event for initiate
      always @(posedge clk or posedge rst) begin
            if (rst == 1)
                  initiate_dly <= 0;
            else
                  initiate_dly <= initiate;
      end
endmodule
```

Synthesis

FPGA Path

The FPGA path was implemented using the DE2-115 board with an Altera FPGA. Quartus II version 13.0 was used for interface with the FPGA. The constraints file used for pin assignments on the development board (to the FPGA) is included in Appendix A.

In migrating code from the original simulations over to FPGA, the major design difference was orienting around physical input. Mainly this meant adapting to push-button switches and making sure to trigger on falling edges of the active low input buttons.

The following figures show the steps for synthesis, place & route, and programming for the FPGA. One can use the compile design button shown in Figure 5.1. This will run all of the steps until programming (one could do each step individually if running into problems). One can then inspect the design summary as shown in Figure 5.2 as well as the chip layout as shown in Fiugre 5.3. To upload your bit file to the FPGA, use the Program Device button at the bottom of Figure 5.1. This will open a new window as shown in Figure 5.4. Here, one uses the JTAG programmer (USB ByteBlaster) to upload program. The FPGA via the development board is connected through a USB cable to a computer as shown in Figure 5.4.

If one is interested in timing, the TimeQuest Timing Analyzer can be used as well. During this project, TimeQuest was used at one point for debugging (although timing was later shown not to be the issue). However, information about TimeQuest usage was not requested in this report.

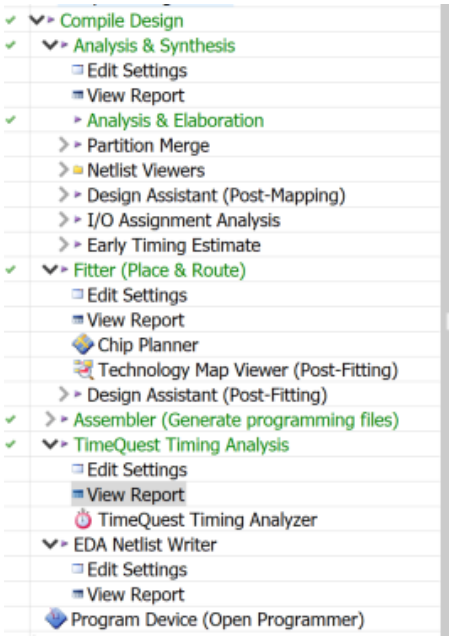

Figure 5.1: Execution steps for FPGA design path in Quartus.

Flow Summary	
Flow Status	Successful - Wed Dec 19 22:53:11 2018
Quartus II 64-Bit Version	13.0.1 Build 232 06...SP 1 SJ Web Edition
Revision Name	daq_top
Top-level Entity Name	daq_top
Family	Cyclone IV E
Device	EP4CE115F29I7
Timing Models	Final
Total logic elements	344 / 114,480 (< 1 %)
Total combinational functions	335 / 114,480 (< 1 %)
Dedicated logic registers	160 / 114,480 (< 1 %)
Total registers	160
Total pins	101 / 529 (19 %)
Total virtual pins	0
Total memory bits	0 / 3,981,312 (0 %)
Embedded Multiplier 9-bit elements	0 / 532 (0 %)
Total PLLs	0 / 4 (0 %)

Figure 5.2: Design report generate from after compilation, synthesis, and place and route.

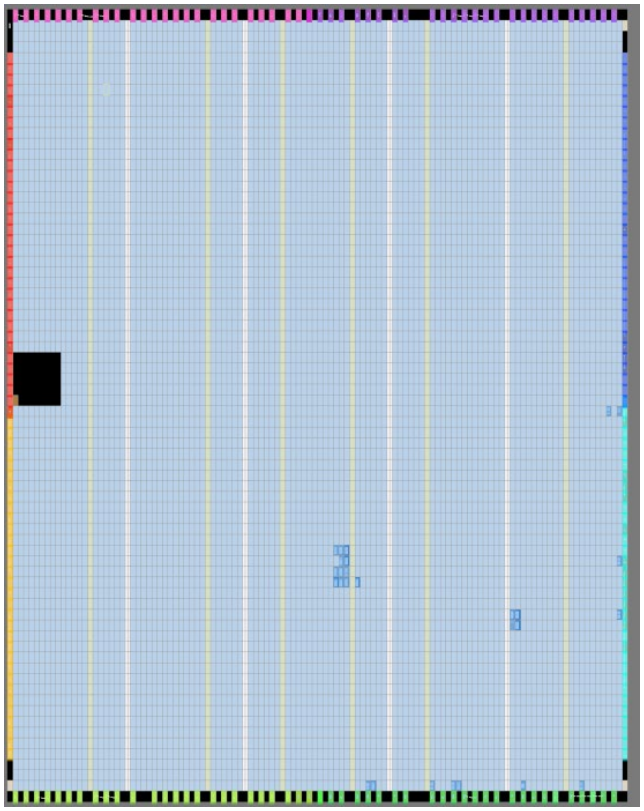

Figure 5.3: Chip layout. Not all that many resources used in this design.

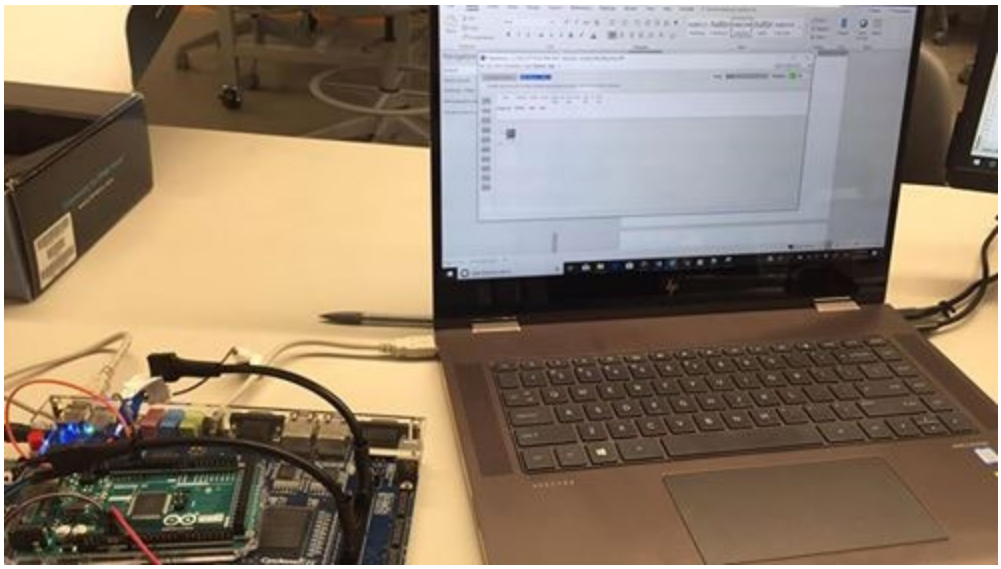

Figure 5.4: How to program the FPGA from Quartus.

ASIC Path

The ASIC path involved synthesizing the verilog code. No major errors were encountered in synthesis. The only warnings shown concerned unsigned numbers, but these were for parameters being used in “if” statements for comparisons.

The timing, area, and power reports are shown in Appendix B. The tcl script used to synthesize and generate reports is included in Appendix A. The full synthesized design at Verilog level is included in Appendix C. The table in Figure 5.5 provides a summary of the device statistics. The remaining figures show the graphical layout of the design. Figure 5.6 is the top block level. Figure 5.7 are the blocks inside the top block. Figure 5.8 is the schematic layout of the daq module (labeled U1) in figure 5.8.

Clock Speed	50 MHz (20ns)
Timing Max Path	Slack 11.28ns (met)
Area	649818 um^2 (not area optimized, units ambiguous on reports)
Dynamic Power	22.06mW
Leakage Power	209.58mW

Figure 5.5: Table of statistics related to ASIC layout. Note, operating voltage of 5V is used for power calculations.

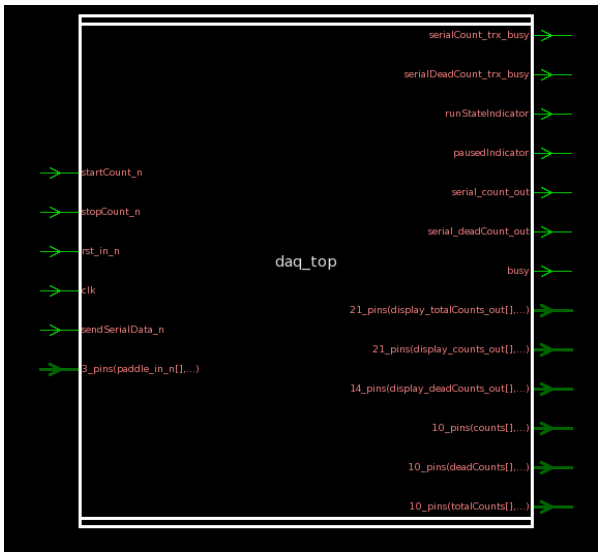

Figure 5.6: Block level representation of design.

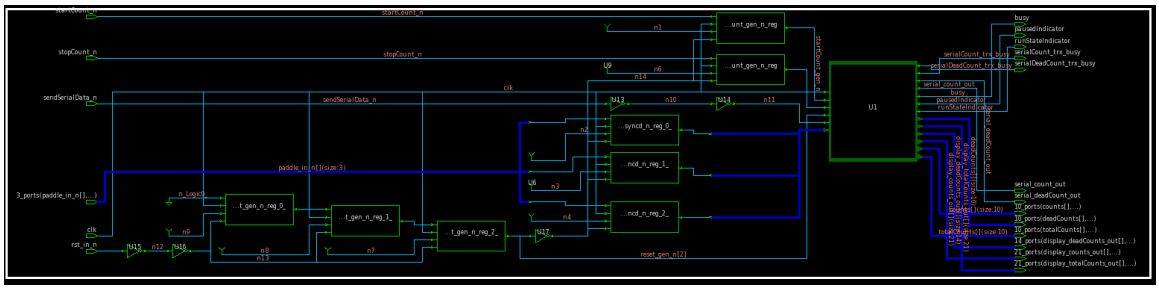

Figure 5.7: Insides of Figure 5.2.

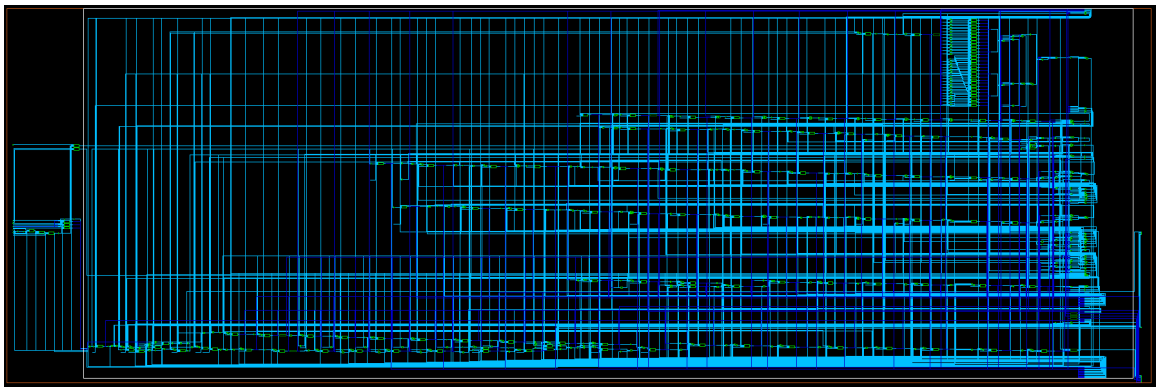

Figure 5.8: Internals of U1 (daq module).

Testing

Functional Testing

Functional testing was done using Cadence simvision. The testbench cycles has multiple modes. The two demonstrated in the below figures first show the basic case where three particles arrive perfectly simultaneously. The second case shows where the third particle (or perhaps third detector) had a delay in the signal arrival but there was still sufficient time coherence with the other two detectors such that a trigger was created. This was done using a random number generator for both the arrival time of the third detector signal and a random number for the width of the pulse generated by the third detector. It would have taken too long to get a hit if all detectors had been random in simulation. In future tests, it would be better to model the arrival times based on probability of arrival of muons in real life.

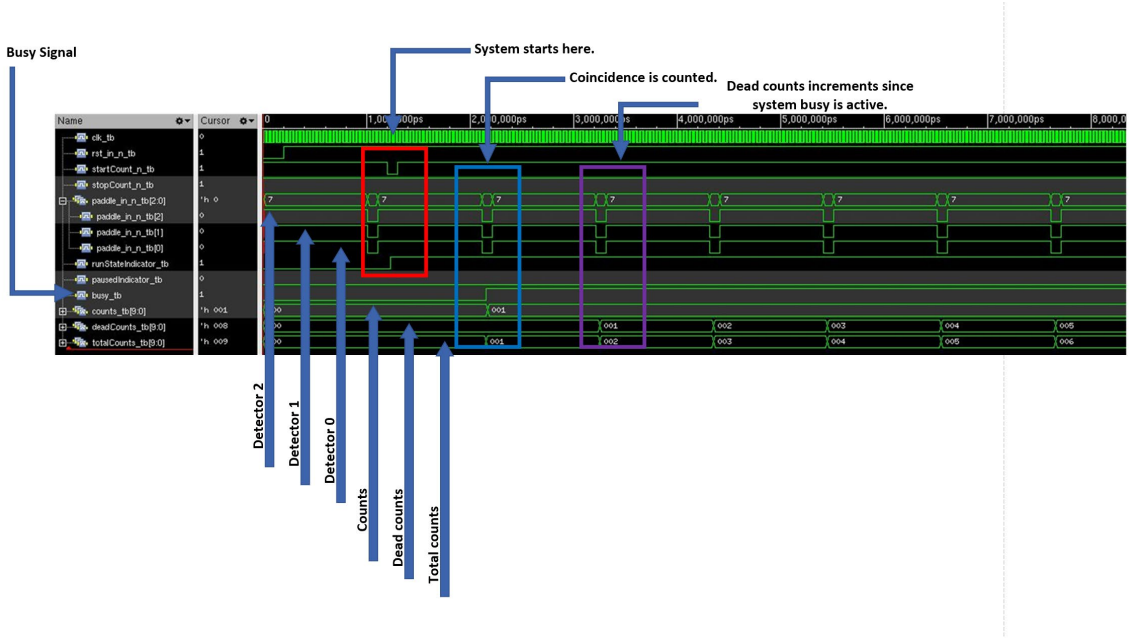

Figure 6.1: The above image demonstrates three particles coming in together.

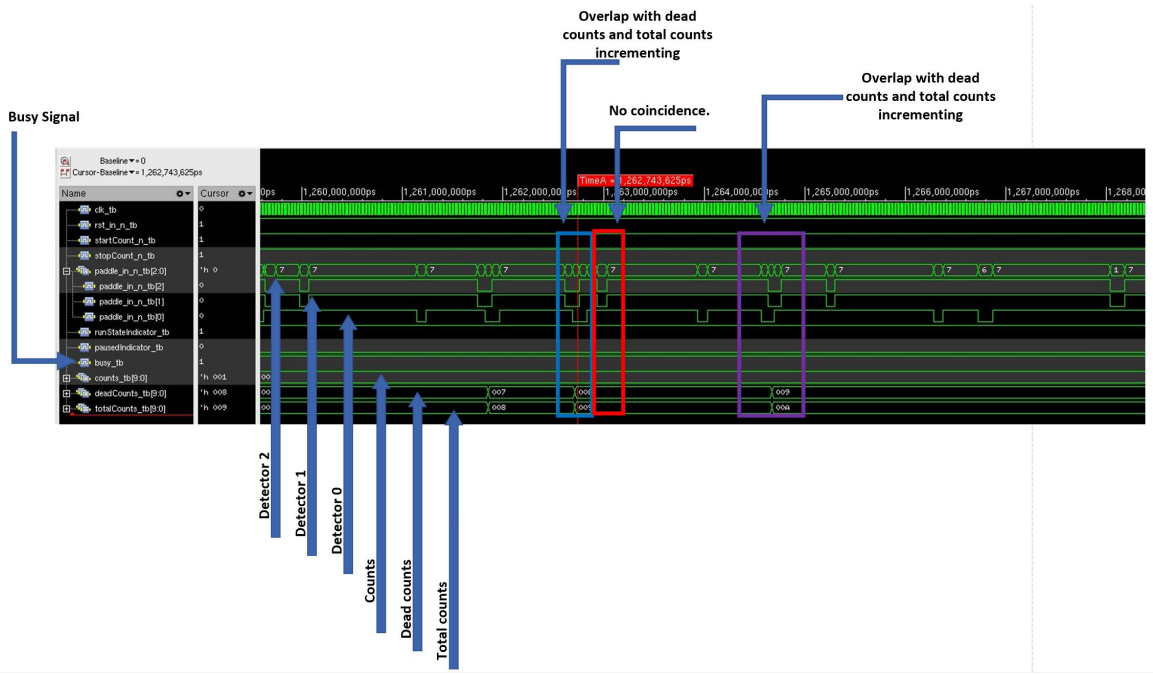

Figure 6.2: This test uses random timing and random pulse width for the third particle.

Testbench Code

```
`timescale 10ns/100ps
module daq_tb_1();
      reg rst_in_n_tb, clk_tb, startCount_n_tb, stopCount_n_tb, sendSerialData_n_tb;
      wire [2:0] paddle_in_n_tb; //input from three sepatate paddles: paddle_0, paddle_1, paddle_2

      wire runStateIndicator_tb, pausedIndicator_tb, busy_tb;
      wire [20:0] display_totalCounts_out_tb, display_counts_out_tb;
      wire [13:0] display_deadCounts_out_tb;
      wire serial_count_out_tb, serial_deadCount_out_tb;
      wire serialCount_trx_busy_tb, serialDeadCount_trx_busy_tb;
      wire[9:0] counts_tb, deadCounts_tb, totalCounts_tb;


      reg[2:0] paddle_in_n_randomTest, paddle_in_n_simulTest;

      reg test_state;
      parameter CONTINUOUS_TRIGGER = 1'b0, RANDOM_PULSES = 1'b1;

      daq_top
u1(.paddle_in_n(paddle_in_n_tb), .startCount_n(startCount_n_tb), .stopCount_n(stopCount_n_tb), .rst_in_n(rs
      t_in_n_tb), .sendSerialData_n(sendSerialData_n_tb), .serialCount_trx_busy(serialCount_trx_busy_tb),
      .serialDeadCount_trx_busy(serialDeadCount_trx_busy_tb), .clk(clk_tb), .runStateIndicator(runStateIndi
      cator_tb), .pausedIndicator(pausedIndicator_tb), .display_totalCounts_out(display_totalCounts_out_tb)
      , .display_counts_out(display_counts_out_tb), .display_deadCounts_out(display_deadCounts_out_tb),
      .serial_count_out(serial_count_out_tb), .serial_deadCount_out(serial_deadCount_out_tb), .counts(count
      s_tb), .deadCounts(deadCounts_tb), .totalCounts(totalCounts_tb), .busy(busy_tb));

      initial begin
            clk_tb = 0;
            startCount_n_tb = 1;
            rst_in_n_tb = 0; //start reset enabled
            stopCount_n_tb = 1;

            //forced test
            /*#20 rst_in_n_tb = 1;
            #50 paddle_in_n_tb = 3'b000;
```

```
      #50 startCount_n_tb = 0;
      #10 startCount_n_tb = 1;
      #100000 stopCount_n_tb = 0;
      #10 stopCount_n_tb = 1;
      #100 rst_in_n_tb = 0;*/

      //run through simulatenous test
      #200 rst_in_n_tb = 1;
      #500 test_state = 0;
      #500 startCount_n_tb = 0;
      #100 startCount_n_tb = 1;
      #1000000 stopCount_n_tb = 0;
      #100 stopCount_n_tb = 1;
      #2000 rst_in_n_tb = 0;

      //run through random pulses
      #200 rst_in_n_tb = 1;
      #500 test_state = 1;
      #500 startCount_n_tb = 0;
      #100 startCount_n_tb = 1;
      #1000000 stopCount_n_tb = 0;
      #100 stopCount_n_tb = 1;
      #1000 rst_in_n_tb = 0;

      #1000 $finish;
end

assign paddle_in_n_tb = test_state ? paddle_in_n_randomTest : paddle_in_n_simulTest;

initial begin
      paddle_in_n_simulTest <= 3'b111;
end

always@(posedge clk_tb)
begin: simultaneous_test
      paddle_in_n_simulTest <= 3'b111;
      #1000 paddle_in_n_simulTest <= 3'b000;
      #100 paddle_in_n_simulTest <= 3'b111; //100 ns pulse width
```

```
	end

	integer a, b, c, d, e, f;
	always@(posedge clk_tb)
	begin: random_test
		a = $random%1000 + 1010; //random number from 10 to 1010.
		b = $random%1000 + 1010;
		c = $random%1000 + 1010;
		d = $random%100 + 100;
		e = $random%100 + 100;
		f = $random%100 + 100;

		paddle_in_n_randomTest <= 3'b111;
		fork
		#a paddle_in_n_randomTest[2] <= 1'b0;
		#a paddle_in_n_randomTest[1] <= 1'b0;
		#c paddle_in_n_randomTest[0] <= 1'b0;
		#(a + d) paddle_in_n_randomTest[2] <= 1'b1;
		#(a + d) paddle_in_n_randomTest[1] <= 1'b1;
		#(c + d) paddle_in_n_randomTest[0] <= 1'b1;
		join

	end


	always
		#10 clk_tb = ~clk_tb; //20ns => 50 MHz

	initial begin
	$shm_open ("daq_tb_1.db");
	$shm_probe (daq_tb_1,"AS");
	$shm_save;
	end


endmodule
```

On Board Testing

The on board testing for this project had to strike a balance between those tests from simulation and what the real system will do for research. Since the actual detectors were not ready for testing, they had to be simulated from outside the FPGA. The first iteration was to make two of the detector inputs come from switches on the development board and then use a push-button switch to model the third detector. This was a proof-of-concept approach and too simplistic.

The updated version of testing used an Arduino to model detector inputs to the system. The Arduino code will be included below. The first version of the Arduino only controlled one detector and had the other detectors active continuously. The second version controlled two detectors and had various input patterns to test functionality of counts, dead counts, and total counts. For the purposes of this last test, the length of the latched trigger was increased so that the busy signal would be visible on the board. This also increased the proportion of dead counts and total counts as counts does not increment when the busy signal is active (i.e., when the system is busy).

Several systems are tested at the same time in these board tests. These are demonstrated in the videos (4 videos) included in the submission folder with this report; however, a picture of the control and output peripherals on the development board is in Figure 6.3. Video 1 shows operating procedure and then along with Video 2 shows the procedure for transmitting a value and checking on the oscilloscope. Videos 3 and 4 show similar procedures but for transmitting dead counts.

The first functionality is the start, stop, and reset for controls over the entire system. Video 1 (labeled as such in the report folder) shows the system starting along with the green LED lighting, the system stopping along with the red LED lighting and the green LED turning off, and the system reseting with all indicator lights turning off. The second functionality is the counting (counts, dead counts, and total counts). Each counter gets its on 7-seg display. As is labeled in the figure, counts has two digits on the far left, dead counts has two digits on the middle display, and total counts get three of the four digits on the right-most display. When counts increments, a red LED indicating a busy signal turns-on for the length of time set in the design for the latched trigger. This indicates the system is busy and means counts will not increment again until the light turns-off. Dead counts and total counts will increment while the system is busy, though. Counts is also represented in binary using red LEDs. This is for easy testing of the next system, serial data transfer. As the video shows, pressing the transmit button will light two green LEDs. These LEDs represent that counts and dead counts are being serially transmitted from GPIO on the development board. The signals are attached to an oscilloscope. They were tested one at a time for lack of oscilloscope probes. If one is probing the counts transmit, one can use the binary display to easily see if the value matches the transmitted value displayed on the oscilloscope. For the purposes of these tests, each bit in the transmission is held for about one sec. A 16-bit word is transmitted with 3 padding bits at the MSBs, followed by the 10-bit count value, followed by 3 padding bits at the LSBs (all transmitted from MSB to LSB). These padding bits are used to identify that a transmission is starting. For the duration of the transmission, the green LED indicators are lit. When one presses the transmit button, the system samples the current value of counts and dead counts, and the system can continue counting but the transmitted value will not be updated.

Video 3 and 4 show that serial transmission of dead counts while Video 1 shows the serial transmission for counts (amounts to connecting oscilloscope probe to proper output). Video 3

shows the value on the oscilloscope as compared to the value on the 7-seg display to confirm that the dead counts value and transmitted value match.

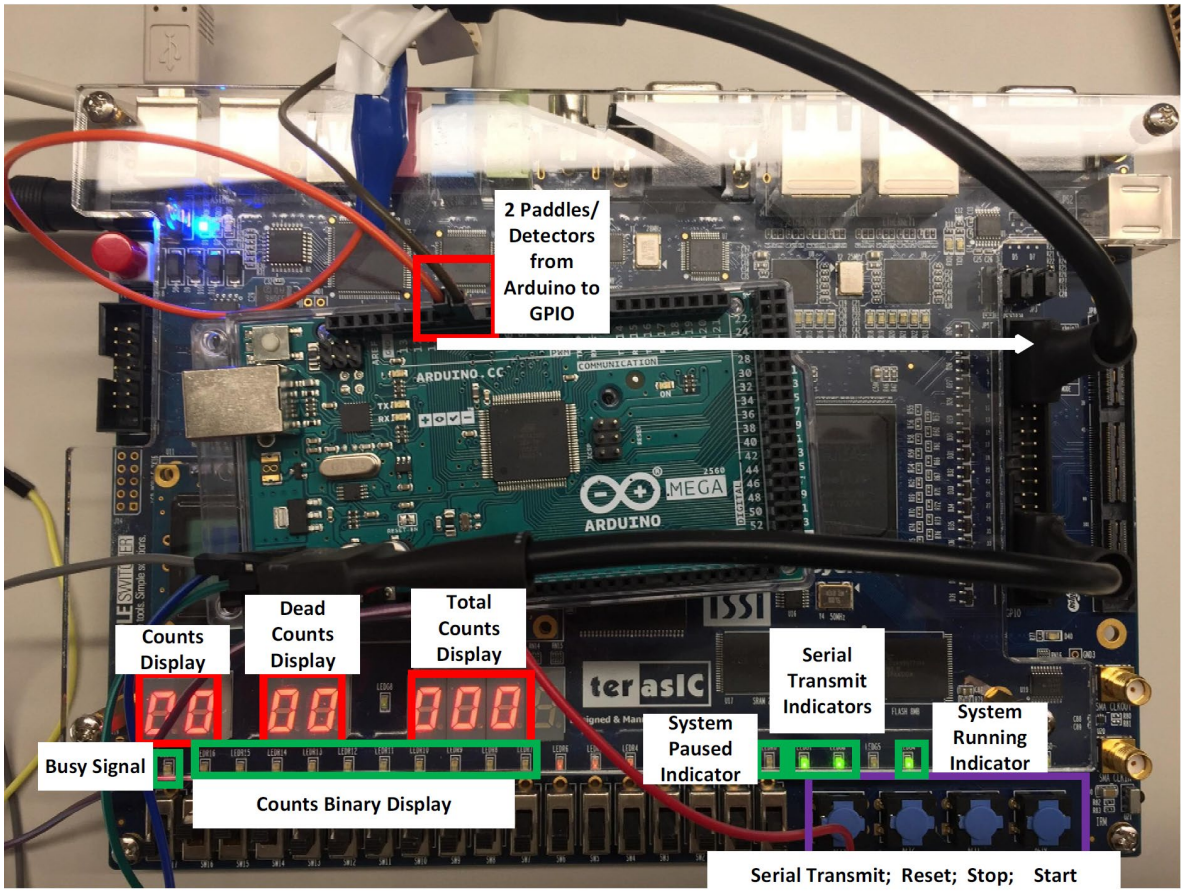

Figure 6.3: FPGA development board with Arduino sitting ontop. User interface and testing components highlighted.

The four videos will make use of the I/O represented in Figure 6.3. To match the buttons pressed in the figure with the expected functionality, reference the figure above.

For all of the Arduino tests, one can tell about how often a count is expected based on the signal delays in the Arduino code. Mentioned in a later section, a better version of testing would have included self-checking to ensure all counts were correct.

Arduino Test Code Version 1

The following code is the simple Arduino testbench. It turns one output from the Arduino on and off to create the active low pulse that the FPGA is looking for. It the low pulse width is 100ms.

```
/*
 This program will generate test signals
 for the DAQ project for ECE6213.
 The signals will represent various PMT paddle signals
*/


int timer = 100;           // The higher the number, the slower the timing.


void setup() {
  // Set mode of three output pins; using analog output pins to control voltage to 2 V:
  pinMode(2, OUTPUT);
  pinMode(9, OUTPUT);
  //pinMode(8, OUTPUT);

  // Set mode for input
  Serial.begin(9600);
}
```

```
void loop() {
    analogWrite(9, 255);
    delay(1000);
    analogWrite(9, 0);
    delay(100);

  delay(1);
}
```

Arduino Test Code Version 2

This version of the Arduino testbench controls two detector inputs to the FPGA. Here, the negative pulse widths vary and at times the two signals are not simultaneous (meaning no trigger and implying that no cosmic particle was incident on the detectors).

```
void setup() {

  // Set mode of three output pins; using analog output pins to control voltage to 2 V:

  pinMode(8, OUTPUT);

  pinMode(9, OUTPUT);


  // Set mode for input

  Serial.begin(9600);

}


void loop() {

    analogWrite(9, 255);

    analogWrite(8, 255);

    delay(1000);


    analogWrite(9, 0);

    analogWrite(8, 0);

    delay(100);
```

```
    analogWrite(9, 255);
    analogWrite(8, 0);
    delay(1000);

    analogWrite(9, 0);
    delay(1);
    analogWrite(8, 255);
    delay(1000);
    analogWrite(8,0);
    delay(10);
    analogWrite(8, 255);

    analogWrite(9, 0);
    delay(100);

  delay(1);
}
```

Problems and Debugging

Fortunately the project did not have too many major errors to debug. The first problem was the latched counter. This cause issues because the counter to keep the latched trigger high needed to be started at a very specific moment. This was accomplished with two separate procedural blocks. One controlled when the trigger would be high and sent a counter start signal. The second would send a signal to the other module when the latch counter reached its terminal value, such that the latched trigger could be released (set low).

The main debugging issue had to do with the counts and dead counts 7-seg displays. The total counts display was working perfectly, so its code was copied for counts and dead counts. When implemented in hardware, counts would reach a value of between 11 and 15 and then would reset back to 10 and then repeat (all while the binary counter for it worked fine). At the same time, dead counts would increment up to 9 and then only the tens digit would increment. Not seeing any typo, I tried isolating various portions of the code, but perhaps was not thorough enough. Then I explored the Timing Analysis tools in Quartus to see if there were any issues there for some reason, but all timing was met. Finally, a pattern was noticed. Once dead counts reached 9, counts would set to 10. There was a typo in the BCD counting, One line was supposed to reference dead counts within the dead counts counter but instead referenced counts, causing these strange dependicies between the two counters. Adding four letters fixed the typo (a copy and paste error, essentially). At least I was able to get some experience with TimeQuest Analyzer.

Conclusions

Overall the project was successful, especially given the timeline and the fact that this was an individual effort. The system is actually ready for real detectors to be connected, but those detectors have not been assembled completely yet (a different student's responsibility).

Given more time, the next step would have been to add buffers at the serial transmit output. Currently, one presses the transmit button and the current counts and dead counts values are sampled and transmission begins. If the user presses transmit again while the system is transmitting, the input will be ignored. Instead, the FIFO designed earlier in the semester could have been used to queue outputs for transmission.

Also, there was not enough time to perfectly perform thorough testing. The original testbench for simulation was decent and used random signal generation to test detector inputs, but self-checking was not designed. The Arduino tests were simple as to show the functionality but were not comprehensive by any means. A nicer version of them could have used self-checking as well to confirm every count is proper.

Originally the system was going to take-in ADC information so that in the real system pulse shape information could be extracted from the detectors (in this report, signals were digital; in the real system they will start-off as analog and be converted; the original analog signals would connect to the ADCs). Complications with choosing the propper ADC circuitry occurred and had to be relegated for later. The new idea is to potentially use development boards for high sample rate ADCs.

Lastly, a new trigger design for the next iteration of this project may be used. The new design would search for a minimum pulse width from the coincidence of the detectors to ensure a signal qualifies as one the system can handle. This is largely outside the scope of the project for this class but related to sample rate for the ADCs and the number of samples that can be attained for pulse widths of different lengths.

Appendix A

TCL Script

```
#####################################################################

#### Design Compiler Script for ECE 6213

#### Performs Synthesis only to AMI .5 technology

#### author: Jack Hirschman taken from Jerry Wu

#### note: this is a TCL script

#####################################################################

###################################

# ITEMS YOU WILL NEED TO SET FOR EACH DESIGN

# 1) myFiles - LIST OF YOUR FILES TO SYNTHESIZE

# 2) basename - TOP LEVEL MODULE IN YOUR DESIGN

# 3) myClk - NAME OF YOUR CLOCK SIGNAL

# 4) virtual - USE A REAL CLOCK (SEQUENTIAL DESIGNS) OR A VIRTUAL

#            CLOCK (COMBINATORIAL DESIGNS)

# 5) myPeriod - SETS THE CLOCK SPEED, THUS DEFINING THE SYNTHESIS SPEED GOAL

###################################


# list of all HDL files in the design

set myFiles [list ./src/daq_top.v ./src/daq.v] ; # modify to your directory/filename
```

```
set basename daq_top;    # Top-level module name; modify as need
set myClk clk;           # The name of your clock; modify as need

set virtual 0 ;          # 1 if virtual clock, 0 if real clock
set myPeriod_ns 20 ;     # desired clock period (in ns) (sets speed goal);aimed at 50MHz

####################################
# Some runtime options, change only if needed
set runname syn                 ;# Name appended to output files
set exit_dc 0                  ;# 1 to exit DC after running, 0 to keep DC running
# set the target library
set target_library [list osu05_stdcells.db] ;
####################################

####################################
# Control the printing of result files
####################################
set verbose 0                  ;# 1 Write reports to screen, 0 do not write reports to screen
set verbose_dft 0       ;# 1 Write reports to screen, 0 do not write reports to screen
####################################
```

```
# Timing and loading information
# Please modify as need; 11/2/15
####################################

set myClkLatency_ns 0.3       ; # clock network latency
set myInDelay_ns 2.0              ;# delay from clock to inputs valid
set myOutDelay_ns 1.65             ;# delay from clock to output valid
set myInputBuf INVX1           ;# name of cell driving the inputs
set myLoadLibrary [file rootname $target_library]    ;# name of library the cell comes from
set myLoadPin A             ;# name of pin that the outputs drive
set myMaxFanout   1     ;# max fanout load for input pins; modify as need
set myOutputLoad 0.1    ;# output pin loading

##################
# compiler switches...
##################
set optimizeArea 0 ;    # 1 for area, 0 for speed
                        # you may switch your optimization for your HW. 11/2/15
set useUltra 0  ; # 1 for compile_ultra, 0 for compile
                                    # mapEffort, useUngroup are for
```

```
                                    # non-ultra compile...
set useUngroup 0                    ;# 0 if no flatten, 1 if flatten

####################################
# Set some system-level things that RARELY change...
####################################
# synthetic_library is set in .synopsys_dc.setup to be
# the dw_foundation library.
set link_library [concat  [concat  "*" $target_library] $synthetic_library]
####################################
set fileFormat verilog              ;# verilog or VHDL

##################################################################
##################################################################
### YOU SHOULD NOT NEED TO CHANGE ANYTHING BELOW THIS LINE ###
##################################################################
##################################################################

##################################################################
#### read in, link to standard cells, and uniquify design ####
```

```
#################################################################

####################################
# remove any other designs from design compiler's memory
####################################
remove_design -all

echo IMPORTING DESIGN
####################################
# analyzer & elaborate verilog source files
####################################
analyze -format $fileFormat -lib WORK $myFiles
elaborate $basename -lib WORK -update

####################################
# set design to 'highest' module level
####################################
current_design $basename

####################################
```

```
# link to standard cell libraries and uniquify
####################################
link
uniquify

#####################################################
#### setup clock & all input/output constraints ####
#####################################################
echo SETTING CONSTRAINTS

####################################
# now you can create clocks for the design
# and set other constraints
####################################
if {  $virtual == 0 } {
   create_clock -period $myPeriod_ns $myClk
} else {
   create_clock -period $myPeriod_ns -name $myClk
}
set_clock_latency $myClkLatency_ns $myClk
```

```
####################################
# set delays on all inputs & outputs with respect to the clock (in ns)
# set the input and output delay relative to myClk
####################################
if {  $virtual == 0 } {
    set_input_delay $myInDelay_ns -clock $myClk [all_inputs]
} else {
    set_input_delay $myInDelay_ns -clock $myClk [remove_from_collection [all_inputs] $myClk]
}
set_output_delay $myOutDelay_ns -clock $myClk [all_outputs]
####################################
# Set the driving cell for all inputs except the clock
# The clock has infinite drive by default. This is usually
# what you want for synthesis because you will use other
# tools (like SOC Encounter) to build the clock tree
# (or define it by hand).
####################################
if {  $virtual == 0 } {
    set_driving_cell  -library $myLoadLibrary -lib_cell $myInputBuf [all_inputs]
} else {
```

```
   set_driving_cell  -library $myLoadLibrary -lib_cell $myInputBuf [remove_from_collection [all_inputs]
$myClk]

}

####################################

# set load/fanin/fanout for all inputs/outputs

####################################

set_load $myOutputLoad [all_outputs]


####################################

# check value of fanout

####################################

set_max_fanout $myMaxFanout [all_inputs]

set_fanout_load 8 [all_outputs]


echo DONE SETTING CONSTRAINTS


####################################

# This command will fix the problem of having

# assign statements left in your structural file.

# But, it will insert pairs of inverters for feedthroughs!

set_fix_multiple_port_nets -all -buffer_constants
```

```
####################################

echo BEGIN COMPILING DESIGN
####################################
# optimize for area
####################################
if { $optimizeArea == 1} {
      set_max_area 0
}
####################################
# now compile the design with given mapping effort
# and do a second compile with incremental mapping
# or use the compile_ultra meta-command
####################################
if {  $useUltra == 1 } {
   compile_ultra
} else {
   if {  $useUngroup == 1 } {
      compile -ungroup_all -map_effort medium
  } else {
```

```
      compile -map_effort medium -exact_map
  }
}
check_design
echo VIOLATIONS
report_constraint -all_violators

#######################################################
#### generate verilog code for synthesized module ###
#### sdc files, sdf files, design compiler project###
#### and write out reports                  ###
#######################################################
echo OUTPUT FILES AND REPORTS
set filebase [format "%s%s" [format "%s%s" $basename "_"] $runname]

####################################
# structural (synthesized) file as verilog
####################################
set filename [format "%s%s%s" ./src/ $filebase ".v"]
redirect change_names { change_names -rules verilog -hierarchy -verbose }
```

```
write -format verilog -hierarchy -output $filename

####################################
# write out the sdf file for back-annotated verilog sim
# This file can be large!
####################################
set filename [format "%s%s%s" ./src/ $filebase ".sdf"]
write_sdf -version 1.0 $filename

####################################
# this is the timing constraints file generated from the
# conditions above - used in the place and route program
####################################
set filename [format "%s%s%s" ./src/ $filebase ".sdc"]
write_sdc $filename

####################################
# generate reports for user to view
####################################
```

```
if { $verbose == 1 } {
      report_design
      report_hierarchy
      report_timing -path full -delay max -nworst 3 -significant_digits 2 -sort_by group
      report_timing -path full -delay min -nworst 3 -significant_digits 2 -sort_by group
      report_area
      report_cell
      report_net
      report_port -v
      report_power -analysis_effort low
}

# Design and Hierarchy reports
set filename [format "%s%s%s" ./reports/ $filebase ".design"]
redirect $filename { report_design }
set filename [format "%s%s%s" ./reports/ $filebase ".design"]
redirect -append $filename { report_hierarchy }

# Timing reports
set filename [format "%s%s%s" ./reports/ $filebase ".timing"]
```

```
redirect $filename { report_timing -path full -delay max -nworst 5 -significant_digits 2 -sort_by group }

set filename [format "%s%s%s" ./reports/ $filebase ".timing"]

redirect -append $filename { report_timing -path full -delay min -nworst 5 -significant_digits 2 -sort_by
group }


# Report_cell and report_area

set filename [format "%s%s%s" ./reports/ $filebase ".area"]

redirect $filename { report_area }

set filename [format "%s%s%s" ./reports/ $filebase ".area"]

redirect -append $filename { report_cell }


# Report port

set filename [format "%s%s%s" ./reports/ $filebase ".ports"]

redirect $filename { report_port -v}


#report net

set filename [format "%s%s%s" ./reports/ $filebase ".net"]

redirect $filename { report_net }


# report power

set filename [format "%s%s%s" ./reports/ $filebase ".pow"]
```

```
redirect $filename { report_power -analysis_effort low }


####################################
# quit dc
####################################
if { $exit_dc == 1} {
      exit
}
```

```
# -------------------------------------------------------------------------- #
#
# Copyright (C) 1991-2013 Altera Corporation
# Your use of Altera Corporation's design tools, logic functions
# and other software and tools, and its AMPP partner logic
# functions, and any output files from any of the foregoing
# (including device programming or simulation files), and any
# associated documentation or information are expressly subject
# to the terms and conditions of the Altera Program License
# Subscription Agreement, Altera MegaCore Function License
# Agreement, or other applicable license agreement, including,
# without limitation, that your use is for the sole purpose of
# programming logic devices manufactured by Altera and sold by
# Altera or its authorized distributors.  Please refer to the
# applicable agreement for further details.
#
# -------------------------------------------------------------------------- #
#
# Quartus II 64-Bit
# Version 13.0.1 Build 232 06/12/2013 Service Pack 1 SJ Web Edition
```

```
# Date created = 23:55:14  November 24, 2018
#
# ------------------------------------------------------------------------- #
#
# Notes:
#
# 1) The default values for assignments are stored in the file:
#           daq_top_assignment_defaults.qdf
#    If this file doesn't exist, see file:
#           assignment_defaults.qdf
#
# 2) Altera recommends that you do not modify this file. This
#    file is updated automatically by the Quartus II software
#    and any changes you make may be lost or overwritten.
#
# ------------------------------------------------------------------------- #


set_global_assignment -name FAMILY "Cyclone IV E"
set_global_assignment -name DEVICE EP4CE115F29I7
```

```
set_global_assignment -name TOP_LEVEL_ENTITY daq_top
set_global_assignment -name ORIGINAL_QUARTUS_VERSION "13.0 SP1"
set_global_assignment -name PROJECT_CREATION_TIME_DATE "23:55:14  NOVEMBER 24, 2018"
set_global_assignment -name LAST_QUARTUS_VERSION "13.0 SP1"
set_global_assignment -name PROJECT_OUTPUT_DIRECTORY output_files
set_global_assignment -name MIN_CORE_JUNCTION_TEMP "-40"
set_global_assignment -name MAX_CORE_JUNCTION_TEMP 100
set_global_assignment -name ERROR_CHECK_FREQUENCY_DIVISOR 1
set_global_assignment -name POWER_PRESET_COOLING_SOLUTION "23 MM HEAT SINK WITH 200 LFPM AIRFLOW"
set_global_assignment -name POWER_BOARD_THERMAL_MODEL "NONE (CONSERVATIVE)"
set_global_assignment -name PARTITION_NETLIST_TYPE SOURCE -section_id Top
set_global_assignment -name PARTITION_FITTER_PRESERVATION_LEVEL PLACEMENT_AND_ROUTING -section_id Top
set_global_assignment -name PARTITION_COLOR 16764057 -section_id Top
set_global_assignment -name STRATIX_DEVICE_IO_STANDARD "2.5 V"
set_location_assignment PIN_M23 -to startCount_n
set_location_assignment PIN_M21 -to stopCount_n
set_location_assignment PIN_N21 -to rst_in_n
set_location_assignment PIN_AC15 -to paddle_in_n[0]
set_location_assignment PIN_AE16 -to paddle_in_n[1]
set_location_assignment PIN_AC28 -to paddle_in_n[2]
```

```
set_location_assignment PIN_Y2 -to clk
set_location_assignment PIN_M24 -to display_totalCounts_out[0]
set_location_assignment PIN_Y19 -to display_totalCounts_out[20]
set_location_assignment PIN_AF23 -to display_totalCounts_out[19]
set_location_assignment PIN_AD24 -to display_totalCounts_out[18]
set_location_assignment PIN_AA21 -to display_totalCounts_out[17]
set_location_assignment PIN_AB20 -to display_totalCounts_out[16]
set_location_assignment PIN_U21 -to display_totalCounts_out[15]
set_location_assignment PIN_W28 -to display_totalCounts_out[13]
set_location_assignment PIN_V21 -to display_totalCounts_out[14]
set_location_assignment PIN_W27 -to display_totalCounts_out[12]
set_location_assignment PIN_Y26 -to display_totalCounts_out[11]
set_location_assignment PIN_W26 -to display_totalCounts_out[10]
set_location_assignment PIN_Y25 -to display_totalCounts_out[9]
set_location_assignment PIN_AA26 -to display_totalCounts_out[8]
set_location_assignment PIN_AA25 -to display_totalCounts_out[7]
set_location_assignment PIN_U24 -to display_totalCounts_out[6]
set_location_assignment PIN_U23 -to display_totalCounts_out[5]
set_location_assignment PIN_W25 -to display_totalCounts_out[4]
set_location_assignment PIN_W22 -to display_totalCounts_out[3]
```

```
set_location_assignment PIN_W21 -to display_totalCounts_out[2]
set_location_assignment PIN_Y22 -to display_totalCounts_out[1]
set_location_assignment PIN_AA14 -to display_counts_out[13]
set_location_assignment PIN_AD17 -to display_counts_out[7]
set_location_assignment PIN_AE17 -to display_counts_out[8]
set_location_assignment PIN_AG17 -to display_counts_out[9]
set_location_assignment PIN_AH17 -to display_counts_out[10]
set_location_assignment PIN_AF17 -to display_counts_out[11]
set_location_assignment PIN_AG18 -to display_counts_out[12]
set_location_assignment PIN_AA17 -to display_counts_out[0]
set_location_assignment PIN_AB16 -to display_counts_out[1]
set_location_assignment PIN_AA16 -to display_counts_out[2]
set_location_assignment PIN_AB17 -to display_counts_out[3]
set_location_assignment PIN_AB15 -to display_counts_out[4]
set_location_assignment PIN_AA15 -to display_counts_out[5]
set_location_assignment PIN_AC17 -to display_counts_out[6]
set_location_assignment PIN_AB19 -to display_deadCounts_out[0]
set_location_assignment PIN_AA19 -to display_deadCounts_out[1]
set_location_assignment PIN_AG21 -to display_deadCounts_out[2]
set_location_assignment PIN_AH21 -to display_deadCounts_out[3]
```

```
set_location_assignment PIN_AE19 -to display_deadCounts_out[4]
set_location_assignment PIN_AF19 -to display_deadCounts_out[5]
set_location_assignment PIN_AE18 -to display_deadCounts_out[6]
set_location_assignment PIN_AD18 -to display_deadCounts_out[7]
set_location_assignment PIN_AC18 -to display_deadCounts_out[8]
set_location_assignment PIN_AB18 -to display_deadCounts_out[9]
set_location_assignment PIN_AH19 -to display_deadCounts_out[10]
set_location_assignment PIN_AG19 -to display_deadCounts_out[11]
set_location_assignment PIN_AF18 -to display_deadCounts_out[12]
set_location_assignment PIN_AH18 -to display_deadCounts_out[13]
set_location_assignment PIN_H21 -to runStateIndicator
set_location_assignment PIN_G19 -to pausedIndicator
set_location_assignment PIN_G22 -to serialCount_trx_busy
set_location_assignment PIN_G21 -to serialDeadCount_trx_busy
set_location_assignment PIN_R24 -to sendSerialData_n
set_location_assignment PIN_AG26 -to serial_count_out
set_location_assignment PIN_AG23 -to serial_deadCount_out
set_location_assignment PIN_H15 -to busy
set_location_assignment PIN_H19 -to counts[0]
set_location_assignment PIN_G16 -to counts[9]
```

```
set_location_assignment PIN_G15 -to counts[8]
set_location_assignment PIN_F15 -to counts[7]
set_location_assignment PIN_H17 -to counts[6]
set_location_assignment PIN_J16 -to counts[5]
set_location_assignment PIN_H16 -to counts[4]
set_location_assignment PIN_J15 -to counts[3]
set_location_assignment PIN_G17 -to counts[2]
set_location_assignment PIN_J17 -to counts[1]
set_global_assignment -name SDC_FILE output_files/daq_top.sdc
set_global_assignment -name SOURCE_FILE atom_netlists/daq_top.qsf
set_global_assignment -name VERILOG_FILE "../../GWU/Fall2018/VLSI 2/Final_Project/src/daq.v"
set_global_assignment -name VERILOG_FILE "../../GWU/Fall2018/VLSI 2/Final_Project/src/daq_top.v"
set_instance_assignment -name PARTITION_HIERARCHY root_partition -to | -section_id To
```

Appendix B

Overview

This appendix includes the synthesis reports from the ASIC path. Synopsys Design Vision was used for synthesis.

Timing Report

```
****************************************
Report : timing
        -path full
        -delay max
        -nworst 5
        -max_paths 5
        -sort_by group
Design : daq_top
Version: O-2018.06-SP1
Date   : Wed Dec 19 13:52:24 2018
****************************************

Operating Conditions: typical   Library: osu05_stdcells
Wire Load Model Mode: top

  Startpoint: U1/latch_counter_reg_1_
```

```
            (rising edge-triggered flip-flop clocked by clk)
Endpoint: U1/latch_counter_reg_29_
          (rising edge-triggered flip-flop clocked by clk)
Path Group: clk
Path Type: max

Point                                     Incr       Path
-----------------------------------------------------------
clock clk (rise edge)                     0.00       0.00
clock network delay (ideal)               0.30       0.30
U1/latch_counter_reg_1_/CLK (DFFSR)       0.00       0.30 r
U1/latch_counter_reg_1_/Q (DFFSR)         0.57       0.87 f
U1/add_99/A[1] (daq_DW01_inc_3)           0.00       0.87 f
U1/add_99/U1_1_1/YC (HAX1)                0.32       1.19 f
U1/add_99/U1_1_2/YC (HAX1)                0.27       1.46 f
U1/add_99/U1_1_3/YC (HAX1)                0.27       1.73 f
U1/add_99/U1_1_4/YC (HAX1)                0.27       2.00 f
U1/add_99/U1_1_5/YC (HAX1)                0.27       2.27 f
U1/add_99/U1_1_6/YC (HAX1)                0.27       2.54 f
U1/add_99/U1_1_7/YC (HAX1)                0.27       2.81 f
```

```
U1/add_99/U1_1_8/YC (HAX1)              0.27       3.08 f
U1/add_99/U1_1_9/YC (HAX1)              0.27       3.34 f
U1/add_99/U1_1_10/YC (HAX1)             0.27       3.61 f
U1/add_99/U1_1_11/YC (HAX1)             0.27       3.88 f
U1/add_99/U1_1_12/YC (HAX1)             0.27       4.15 f
U1/add_99/U1_1_13/YC (HAX1)             0.27       4.42 f
U1/add_99/U1_1_14/YC (HAX1)             0.27       4.69 f
U1/add_99/U1_1_15/YC (HAX1)             0.27       4.96 f
U1/add_99/U1_1_16/YC (HAX1)             0.27       5.23 f
U1/add_99/U1_1_17/YC (HAX1)             0.27       5.50 f
U1/add_99/U1_1_18/YC (HAX1)             0.27       5.77 f
U1/add_99/U1_1_19/YC (HAX1)             0.27       6.04 f
U1/add_99/U1_1_20/YC (HAX1)             0.27       6.31 f
U1/add_99/U1_1_21/YC (HAX1)             0.27       6.58 f
U1/add_99/U1_1_22/YC (HAX1)             0.27       6.85 f
U1/add_99/U1_1_23/YC (HAX1)             0.27       7.12 f
U1/add_99/U1_1_24/YC (HAX1)             0.27       7.39 f
U1/add_99/U1_1_25/YC (HAX1)             0.27       7.66 f
U1/add_99/U1_1_26/YC (HAX1)             0.27       7.93 f
U1/add_99/U1_1_27/YC (HAX1)             0.27       8.20 f
```

```
U1/add_99/U1_1_28/YC (HAX1)              0.31       8.51 f
U1/add_99/U1/Y (XOR2X1)                  0.16       8.67 r
U1/add_99/SUM[29] (daq_DW01_inc_3)       0.00       8.67 r
U1/U138/Y (AND2X2)                       0.14       8.81 r
U1/latch_counter_reg_29_/D (DFFSR)       0.00       8.81 r
data arrival time                                   8.81

clock clk (rise edge)                   20.00      20.00
clock network delay (ideal)              0.30      20.30
U1/latch_counter_reg_29_/CLK (DFFSR)     0.00      20.30 r
library setup time                      -0.21      20.09
data required time                                 20.09
-----------------------------------------------------------
data required time                                 20.09
data arrival time                                  -8.81
-----------------------------------------------------------
slack (MET)                                        11.28


Startpoint: U1/latch_counter_reg_0_
```

```
            (rising edge-triggered flip-flop clocked by clk)
Endpoint: U1/latch_counter_reg_29_
          (rising edge-triggered flip-flop clocked by clk)
Path Group: clk
Path Type: max

Point                                     Incr       Path
-----------------------------------------------------------
clock clk (rise edge)                     0.00       0.00
clock network delay (ideal)               0.30       0.30
U1/latch_counter_reg_0_/CLK (DFFSR)       0.00       0.30 r
U1/latch_counter_reg_0_/Q (DFFSR)         0.59       0.89 f
U1/add_99/A[0] (daq_DW01_inc_3)           0.00       0.89 f
U1/add_99/U1_1_1/YC (HAX1)                0.30       1.18 f
U1/add_99/U1_1_2/YC (HAX1)                0.27       1.45 f
U1/add_99/U1_1_3/YC (HAX1)                0.27       1.72 f
U1/add_99/U1_1_4/YC (HAX1)                0.27       1.99 f
U1/add_99/U1_1_5/YC (HAX1)                0.27       2.26 f
U1/add_99/U1_1_6/YC (HAX1)                0.27       2.53 f
U1/add_99/U1_1_7/YC (HAX1)                0.27       2.80 f
```

```
U1/add_99/U1_1_8/YC (HAX1)              0.27       3.07 f
U1/add_99/U1_1_9/YC (HAX1)              0.27       3.34 f
U1/add_99/U1_1_10/YC (HAX1)             0.27       3.61 f
U1/add_99/U1_1_11/YC (HAX1)             0.27       3.88 f
U1/add_99/U1_1_12/YC (HAX1)             0.27       4.15 f
U1/add_99/U1_1_13/YC (HAX1)             0.27       4.42 f
U1/add_99/U1_1_14/YC (HAX1)             0.27       4.69 f
U1/add_99/U1_1_15/YC (HAX1)             0.27       4.96 f
U1/add_99/U1_1_16/YC (HAX1)             0.27       5.23 f
U1/add_99/U1_1_17/YC (HAX1)             0.27       5.50 f
U1/add_99/U1_1_18/YC (HAX1)             0.27       5.77 f
U1/add_99/U1_1_19/YC (HAX1)             0.27       6.04 f
U1/add_99/U1_1_20/YC (HAX1)             0.27       6.31 f
U1/add_99/U1_1_21/YC (HAX1)             0.27       6.58 f
U1/add_99/U1_1_22/YC (HAX1)             0.27       6.84 f
U1/add_99/U1_1_23/YC (HAX1)             0.27       7.11 f
U1/add_99/U1_1_24/YC (HAX1)             0.27       7.38 f
U1/add_99/U1_1_25/YC (HAX1)             0.27       7.65 f
U1/add_99/U1_1_26/YC (HAX1)             0.27       7.92 f
U1/add_99/U1_1_27/YC (HAX1)             0.27       8.19 f
```

```
U1/add_99/U1_1_28/YC (HAX1)              0.31       8.50 f
U1/add_99/U1/Y (XOR2X1)                  0.16       8.66 r
U1/add_99/SUM[29] (daq_DW01_inc_3)       0.00       8.66 r
U1/U138/Y (AND2X2)                       0.14       8.81 r
U1/latch_counter_reg_29_/D (DFFSR)       0.00       8.81 r
data arrival time                                   8.81

clock clk (rise edge)                   20.00      20.00
clock network delay (ideal)              0.30      20.30
U1/latch_counter_reg_29_/CLK (DFFSR)     0.00      20.30 r
library setup time                      -0.21      20.09
data required time                                 20.09
-----------------------------------------------------------
data required time                                 20.09
data arrival time                                  -8.81
-----------------------------------------------------------
slack (MET)                                        11.28


Startpoint: U1/serialCountOut/trx_delay_counter_reg_1_
```

```
            (rising edge-triggered flip-flop clocked by clk)
Endpoint: U1/serialCountOut/trx_delay_counter_reg_29_
          (rising edge-triggered flip-flop clocked by clk)
Path Group: clk
Path Type: max

Point                                                    Incr       Path
-----------------------------------------------------------------------------
clock clk (rise edge)                                    0.00       0.00
clock network delay (ideal)                              0.30       0.30
U1/serialCountOut/trx_delay_counter_reg_1_/CLK (DFFSR)
                                                         0.00       0.30 r
U1/serialCountOut/trx_delay_counter_reg_1_/Q (DFFSR)
                                                         0.52       0.82 f
U1/serialCountOut/r136/A[1] (serialOut_1_DW01_inc_0)
                                                         0.00       0.82 f
U1/serialCountOut/r136/U1_1_1/YC (HAX1)                  0.30       1.12 f
U1/serialCountOut/r136/U1_1_2/YC (HAX1)                  0.27       1.39 f
U1/serialCountOut/r136/U1_1_3/YC (HAX1)                  0.27       1.66 f
U1/serialCountOut/r136/U1_1_4/YC (HAX1)                  0.27       1.93 f
```

```
U1/serialCountOut/r136/U1_1_5/YC (HAX1)                 0.27       2.20 f
U1/serialCountOut/r136/U1_1_6/YC (HAX1)                 0.27       2.47 f
U1/serialCountOut/r136/U1_1_7/YC (HAX1)                 0.27       2.73 f
U1/serialCountOut/r136/U1_1_8/YC (HAX1)                 0.27       3.01 f
U1/serialCountOut/r136/U1_1_9/YC (HAX1)                 0.27       3.28 f
U1/serialCountOut/r136/U1_1_10/YC (HAX1)                0.27       3.54 f
U1/serialCountOut/r136/U1_1_11/YC (HAX1)                0.27       3.82 f
U1/serialCountOut/r136/U1_1_12/YC (HAX1)                0.27       4.09 f
U1/serialCountOut/r136/U1_1_13/YC (HAX1)                0.27       4.36 f
U1/serialCountOut/r136/U1_1_14/YC (HAX1)                0.27       4.63 f
U1/serialCountOut/r136/U1_1_15/YC (HAX1)                0.27       4.90 f
U1/serialCountOut/r136/U1_1_16/YC (HAX1)                0.27       5.17 f
U1/serialCountOut/r136/U1_1_17/YC (HAX1)                0.27       5.45 f
U1/serialCountOut/r136/U1_1_18/YC (HAX1)                0.27       5.72 f
U1/serialCountOut/r136/U1_1_19/YC (HAX1)                0.27       5.99 f
U1/serialCountOut/r136/U1_1_20/YC (HAX1)                0.27       6.26 f
U1/serialCountOut/r136/U1_1_21/YC (HAX1)                0.27       6.53 f
U1/serialCountOut/r136/U1_1_22/YC (HAX1)                0.27       6.80 f
U1/serialCountOut/r136/U1_1_23/YC (HAX1)                0.27       7.08 f
U1/serialCountOut/r136/U1_1_24/YC (HAX1)                0.27       7.35 f
```

```
U1/serialCountOut/r136/U1_1_25/YC (HAX1)                0.27       7.62 f
U1/serialCountOut/r136/U1_1_26/YC (HAX1)                0.27       7.89 f
U1/serialCountOut/r136/U1_1_27/YC (HAX1)                0.27       8.16 f
U1/serialCountOut/r136/U1_1_28/YC (HAX1)                0.31       8.47 f
U1/serialCountOut/r136/U1/Y (XOR2X1)                    0.16       8.63 r
U1/serialCountOut/r136/SUM[29] (serialOut_1_DW01_inc_0)
                                                        0.00       8.63 r
U1/serialCountOut/U12/Y (AND2X1)                        0.12       8.75 r
U1/serialCountOut/trx_delay_counter_reg_29_/D (DFFSR)
                                                        0.00       8.75 r
data arrival time                                                  8.75

clock clk (rise edge)                                  20.00      20.00
clock network delay (ideal)                             0.30      20.30
U1/serialCountOut/trx_delay_counter_reg_29_/CLK (DFFSR)
                                                        0.00      20.30 r
library setup time                                     -0.22      20.08
data required time                                                20.08
-----------------------------------------------------------------------
data required time                                                20.08
```

```
data arrival time                                                  -8.75
-------------------------------------------------------------------------
slack (MET)                                                        11.33


Startpoint: U1/serialDeadCountOut/trx_delay_counter_reg_1_
            (rising edge-triggered flip-flop clocked by clk)
Endpoint: U1/serialDeadCountOut/trx_delay_counter_reg_29_
          (rising edge-triggered flip-flop clocked by clk)
Path Group: clk
Path Type: max

Point                                                 Incr       Path
-------------------------------------------------------------------------
clock clk (rise edge)                                 0.00       0.00
clock network delay (ideal)                           0.30       0.30
U1/serialDeadCountOut/trx_delay_counter_reg_1_/CLK (DFFSR)
                                                      0.00       0.30 r
U1/serialDeadCountOut/trx_delay_counter_reg_1_/Q (DFFSR)
                                                      0.52       0.82 f
```

```
U1/serialDeadCountOut/r136/A[1] (serialOut_0_DW01_inc_0)
                                                  0.00       0.82 f
U1/serialDeadCountOut/r136/U1_1_1/YC (HAX1)       0.30       1.12 f
U1/serialDeadCountOut/r136/U1_1_2/YC (HAX1)       0.27       1.39 f
U1/serialDeadCountOut/r136/U1_1_3/YC (HAX1)       0.27       1.66 f
U1/serialDeadCountOut/r136/U1_1_4/YC (HAX1)       0.27       1.93 f
U1/serialDeadCountOut/r136/U1_1_5/YC (HAX1)       0.27       2.20 f
U1/serialDeadCountOut/r136/U1_1_6/YC (HAX1)       0.27       2.47 f
U1/serialDeadCountOut/r136/U1_1_7/YC (HAX1)       0.27       2.73 f
U1/serialDeadCountOut/r136/U1_1_8/YC (HAX1)       0.27       3.01 f
U1/serialDeadCountOut/r136/U1_1_9/YC (HAX1)       0.27       3.28 f
U1/serialDeadCountOut/r136/U1_1_10/YC (HAX1)      0.27       3.54 f
U1/serialDeadCountOut/r136/U1_1_11/YC (HAX1)      0.27       3.82 f
U1/serialDeadCountOut/r136/U1_1_12/YC (HAX1)      0.27       4.09 f
U1/serialDeadCountOut/r136/U1_1_13/YC (HAX1)      0.27       4.36 f
U1/serialDeadCountOut/r136/U1_1_14/YC (HAX1)      0.27       4.63 f
U1/serialDeadCountOut/r136/U1_1_15/YC (HAX1)      0.27       4.90 f
U1/serialDeadCountOut/r136/U1_1_16/YC (HAX1)      0.27       5.17 f
U1/serialDeadCountOut/r136/U1_1_17/YC (HAX1)      0.27       5.45 f
U1/serialDeadCountOut/r136/U1_1_18/YC (HAX1)      0.27       5.72 f
```

```
U1/serialDeadCountOut/r136/U1_1_19/YC (HAX1)            0.27       5.99 f
U1/serialDeadCountOut/r136/U1_1_20/YC (HAX1)            0.27       6.26 f
U1/serialDeadCountOut/r136/U1_1_21/YC (HAX1)            0.27       6.53 f
U1/serialDeadCountOut/r136/U1_1_22/YC (HAX1)            0.27       6.80 f
U1/serialDeadCountOut/r136/U1_1_23/YC (HAX1)            0.27       7.08 f
U1/serialDeadCountOut/r136/U1_1_24/YC (HAX1)            0.27       7.35 f
U1/serialDeadCountOut/r136/U1_1_25/YC (HAX1)            0.27       7.62 f
U1/serialDeadCountOut/r136/U1_1_26/YC (HAX1)            0.27       7.89 f
U1/serialDeadCountOut/r136/U1_1_27/YC (HAX1)            0.27       8.16 f
U1/serialDeadCountOut/r136/U1_1_28/YC (HAX1)            0.31       8.47 f
U1/serialDeadCountOut/r136/U1/Y (XOR2X1)                0.16       8.63 r
U1/serialDeadCountOut/r136/SUM[29] (serialOut_0_DW01_inc_0)
                                                        0.00       8.63 r
U1/serialDeadCountOut/U12/Y (AND2X1)                    0.12       8.75 r
U1/serialDeadCountOut/trx_delay_counter_reg_29_/D (DFFSR)
                                                        0.00       8.75 r
data arrival time                                                  8.75

clock clk (rise edge)                                  20.00      20.00
clock network delay (ideal)                             0.30      20.30
```

```
U1/serialDeadCountOut/trx_delay_counter_reg_29_/CLK (DFFSR)
                                                    0.00      20.30 r
library setup time                                 -0.22      20.08
data required time                                            20.08
-------------------------------------------------------------------------
data required time                                            20.08
data arrival time                                             -8.75
-------------------------------------------------------------------------
slack (MET)                                                   11.33


Startpoint: U1/serialCountOut/trx_delay_counter_reg_0_
            (rising edge-triggered flip-flop clocked by clk)
Endpoint: U1/serialCountOut/trx_delay_counter_reg_29_
          (rising edge-triggered flip-flop clocked by clk)
Path Group: clk
Path Type: max

Point                                              Incr       Path
-------------------------------------------------------------------------
```

```
clock clk (rise edge)                                  0.00       0.00
clock network delay (ideal)                            0.30       0.30
U1/serialCountOut/trx_delay_counter_reg_0_/CLK (DFFSR)
                                                       0.00       0.30 r
U1/serialCountOut/trx_delay_counter_reg_0_/Q (DFFSR)
                                                       0.54       0.84 f
U1/serialCountOut/r136/A[0] (serialOut_1_DW01_inc_0)
                                                       0.00       0.84 f
U1/serialCountOut/r136/U1_1_1/YC (HAX1)                0.28       1.12 f
U1/serialCountOut/r136/U1_1_2/YC (HAX1)                0.27       1.39 f
U1/serialCountOut/r136/U1_1_3/YC (HAX1)                0.27       1.65 f
U1/serialCountOut/r136/U1_1_4/YC (HAX1)                0.27       1.92 f
U1/serialCountOut/r136/U1_1_5/YC (HAX1)                0.27       2.19 f
U1/serialCountOut/r136/U1_1_6/YC (HAX1)                0.27       2.46 f
U1/serialCountOut/r136/U1_1_7/YC (HAX1)                0.27       2.73 f
U1/serialCountOut/r136/U1_1_8/YC (HAX1)                0.27       3.00 f
U1/serialCountOut/r136/U1_1_9/YC (HAX1)                0.27       3.27 f
U1/serialCountOut/r136/U1_1_10/YC (HAX1)               0.27       3.54 f
U1/serialCountOut/r136/U1_1_11/YC (HAX1)               0.27       3.81 f
U1/serialCountOut/r136/U1_1_12/YC (HAX1)               0.27       4.09 f
```

```
U1/serialCountOut/r136/U1_1_13/YC (HAX1)                0.27       4.36 f
U1/serialCountOut/r136/U1_1_14/YC (HAX1)                0.27       4.63 f
U1/serialCountOut/r136/U1_1_15/YC (HAX1)                0.27       4.90 f
U1/serialCountOut/r136/U1_1_16/YC (HAX1)                0.27       5.17 f
U1/serialCountOut/r136/U1_1_17/YC (HAX1)                0.27       5.44 f
U1/serialCountOut/r136/U1_1_18/YC (HAX1)                0.27       5.72 f
U1/serialCountOut/r136/U1_1_19/YC (HAX1)                0.27       5.99 f
U1/serialCountOut/r136/U1_1_20/YC (HAX1)                0.27       6.26 f
U1/serialCountOut/r136/U1_1_21/YC (HAX1)                0.27       6.53 f
U1/serialCountOut/r136/U1_1_22/YC (HAX1)                0.27       6.80 f
U1/serialCountOut/r136/U1_1_23/YC (HAX1)                0.27       7.07 f
U1/serialCountOut/r136/U1_1_24/YC (HAX1)                0.27       7.34 f
U1/serialCountOut/r136/U1_1_25/YC (HAX1)                0.27       7.61 f
U1/serialCountOut/r136/U1_1_26/YC (HAX1)                0.27       7.89 f
U1/serialCountOut/r136/U1_1_27/YC (HAX1)                0.27       8.16 f
U1/serialCountOut/r136/U1_1_28/YC (HAX1)                0.31       8.47 f
U1/serialCountOut/r136/U1/Y (XOR2X1)                    0.16       8.63 r
U1/serialCountOut/r136/SUM[29] (serialOut_1_DW01_inc_0)
                                                        0.00       8.63 r
U1/serialCountOut/U12/Y (AND2X1)                        0.12       8.75 r
```

```
  U1/serialCountOut/trx_delay_counter_reg_29_/D (DFFSR)
                                                  0.00       8.75 r
  data arrival time                                          8.75

  clock clk (rise edge)                          20.00      20.00
  clock network delay (ideal)                     0.30      20.30
  U1/serialCountOut/trx_delay_counter_reg_29_/CLK (DFFSR)
                                                  0.00      20.30 r
  library setup time                             -0.22      20.08
  data required time                                        20.08
  --------------------------------------------------------------------------
  data required time                                        20.08
  data arrival time                                         -8.75
  --------------------------------------------------------------------------
  slack (MET)                                               11.33


1


****************************************
```

```
Report : timing
        -path full
        -delay min
        -nworst 5
        -max_paths 5
        -sort_by group
Design : daq_top
Version: O-2018.06-SP1
Date   : Wed Dec 19 13:52:24 2018
****************************************

Operating Conditions: typical   Library: osu05_stdcells
Wire Load Model Mode: top

  Startpoint: reset_gen_n_reg_0_/CLK
              (internal path startpoint clocked by clk)
  Endpoint: reset_gen_n_reg_1_
            (rising edge-triggered flip-flop clocked by clk)
  Path Group: clk
  Path Type: min
```

```
Point                                   Incr       Path
-------------------------------------------------------------
clock clk (rise edge)                   0.00       0.00
clock network delay (ideal)             0.30       0.30
input external delay                    0.00       0.30 r
reset_gen_n_reg_0_/CLK (DFFSR)          0.00       0.30 r
reset_gen_n_reg_0_/Q (DFFSR)            0.45       0.75 f
reset_gen_n_reg_1_/D (DFFSR)            0.00       0.75 f
data arrival time                                  0.75

clock clk (rise edge)                   0.00       0.00
clock network delay (ideal)             0.30       0.30
reset_gen_n_reg_1_/CLK (DFFSR)          0.00       0.30 r
library hold time                       0.10       0.40
data required time                                 0.40
-------------------------------------------------------------
data required time                                 0.40
data arrival time                                 -0.75
-------------------------------------------------------------
```

```
slack (MET)                                          0.36


Startpoint: reset_gen_n_reg_1_
            (rising edge-triggered flip-flop clocked by clk)
Endpoint: reset_gen_n_reg_2_
          (rising edge-triggered flip-flop clocked by clk)
Path Group: clk
Path Type: min

Point                                    Incr       Path
-------------------------------------------------------------
clock clk (rise edge)                    0.00       0.00
clock network delay (ideal)              0.30       0.30
reset_gen_n_reg_1_/CLK (DFFSR)           0.00       0.30 r
reset_gen_n_reg_1_/Q (DFFSR)             0.45       0.75 f
reset_gen_n_reg_2_/D (DFFSR)             0.00       0.75 f
data arrival time                                   0.75

clock clk (rise edge)                    0.00       0.00
```

```
clock network delay (ideal)               0.30       0.30
reset_gen_n_reg_2_/CLK (DFFSR)            0.00       0.30 r
library hold time                         0.10       0.40
data required time                                   0.40
-----------------------------------------------------------
data required time                                   0.40
data arrival time                                   -0.75
-----------------------------------------------------------
slack (MET)                                          0.36


Startpoint: reset_gen_n_reg_0_/CLK
            (internal path startpoint clocked by clk)
Endpoint: reset_gen_n_reg_1_
          (rising edge-triggered flip-flop clocked by clk)
Path Group: clk
Path Type: min

Point                                     Incr       Path
-----------------------------------------------------------
```

```
clock clk (rise edge)                    0.00       0.00
clock network delay (ideal)              0.30       0.30
input external delay                     0.00       0.30 r
reset_gen_n_reg_0_/CLK (DFFSR)           0.00       0.30 r
reset_gen_n_reg_0_/Q (DFFSR)             0.41       0.71 r
reset_gen_n_reg_1_/D (DFFSR)             0.00       0.71 r
data arrival time                                   0.71

clock clk (rise edge)                    0.00       0.00
clock network delay (ideal)              0.30       0.30
reset_gen_n_reg_1_/CLK (DFFSR)           0.00       0.30 r
library hold time                        0.00       0.30
data required time                                  0.30
-----------------------------------------------------------
data required time                                  0.30
data arrival time                                  -0.71
-----------------------------------------------------------
slack (MET)                                         0.41
```

```
Startpoint: reset_gen_n_reg_1_
            (rising edge-triggered flip-flop clocked by clk)
Endpoint: reset_gen_n_reg_2_
          (rising edge-triggered flip-flop clocked by clk)
Path Group: clk
Path Type: min

Point                                    Incr       Path
-----------------------------------------------------------
clock clk (rise edge)                    0.00       0.00
clock network delay (ideal)              0.30       0.30
reset_gen_n_reg_1_/CLK (DFFSR)           0.00       0.30 r
reset_gen_n_reg_1_/Q (DFFSR)             0.41       0.71 r
reset_gen_n_reg_2_/D (DFFSR)             0.00       0.71 r
data arrival time                                   0.71

clock clk (rise edge)                    0.00       0.00
clock network delay (ideal)              0.30       0.30
reset_gen_n_reg_2_/CLK (DFFSR)           0.00       0.30 r
library hold time                        0.00       0.30
```

```
data required time                                         0.30
-----------------------------------------------------------
data required time                                         0.30
data arrival time                                         -0.71
-----------------------------------------------------------
slack (MET)                                                0.41


Startpoint: stopCount_gen_n_reg
            (rising edge-triggered flip-flop clocked by clk)
Endpoint: U1/stopCount_n_dly_reg
          (rising edge-triggered flip-flop clocked by clk)
Path Group: clk
Path Type: min

Point                                      Incr       Path
-----------------------------------------------------------
clock clk (rise edge)                      0.00       0.00
clock network delay (ideal)                0.30       0.30
stopCount_gen_n_reg/CLK (DFFSR)            0.00       0.30 r
```

```
  stopCount_gen_n_reg/Q (DFFSR)             0.57       0.87 f
  U1/stopCount_n (daq)                      0.00       0.87 f
  U1/stopCount_n_dly_reg/D (DFFSR)          0.00       0.87 f
  data arrival time                                    0.87

  clock clk (rise edge)                     0.00       0.00
  clock network delay (ideal)               0.30       0.30
  U1/stopCount_n_dly_reg/CLK (DFFSR)        0.00       0.30 r
  library hold time                         0.06       0.36
  data required time                                   0.36
  -----------------------------------------------------------
  data required time                                   0.36
  data arrival time                                   -0.87
  -----------------------------------------------------------
  slack (MET)                                          0.51


1
```

Area Report

```
****************************************
Report : area
Design : daq_top
Version: O-2018.06-SP1
Date   : Wed Dec 19 13:52:24 2018
****************************************


Library(s) Used:

    osu05_stdcells (File: /apps/design_kits/osu_stdcells_v2p7/synopsys/lib/ami05/osu05_stdcells.db)


Number of ports:                          560
Number of nets:                          2180
Number of cells:                         1617
Number of combinational cells:           1202
Number of sequential cells:               403
Number of macros/black boxes:               0
Number of buf/inv:                        282
Number of references:                       3
```

```
Combinational area:              331434.000000
Buf/Inv area:                     41976.000000
Noncombinational area:           318384.000000
Macro/Black Box area:                 0.000000
Net Interconnect area:      undefined  (No wire load specified)

Total cell area:                 649818.000000
Total area:                 undefined
1

****************************************
Report : cell
Design : daq_top
Version: O-2018.06-SP1
Date   : Wed Dec 19 13:52:24 2018
****************************************

Attributes:
    b - black box (unknown)
    h - hierarchical
```

```
    n - noncombinational
    r - removable
    u - contains unmapped logic

Cell                      Reference       Library             Area  Attributes
------------------------------------------------------------------------------
U1                        daq                              636426.000000
                                                                     h, n
U13                       INVX2           osu05_stdcells   144.000000
U14                       INVX2           osu05_stdcells   144.000000
U15                       INVX2           osu05_stdcells   144.000000
U16                       INVX2           osu05_stdcells   144.000000
U17                       INVX2           osu05_stdcells   144.000000
paddle_in_syncd_n_reg_0_  DFFSR           osu05_stdcells  1584.000000
                                                                     n
paddle_in_syncd_n_reg_1_  DFFSR           osu05_stdcells  1584.000000
                                                                     n
paddle_in_syncd_n_reg_2_  DFFSR           osu05_stdcells  1584.000000
                                                                     n
reset_gen_n_reg_0_        DFFSR           osu05_stdcells  1584.000000
```

```
                                                                        n
reset_gen_n_reg_1_        DFFSR           osu05_stdcells  1584.000000
                                                                        n
reset_gen_n_reg_2_        DFFSR           osu05_stdcells  1584.000000
                                                                        n
startCount_gen_n_reg      DFFSR           osu05_stdcells  1584.000000
                                                                        n
stopCount_gen_n_reg       DFFSR           osu05_stdcells  1584.000000
                                                                        n
--------------------------------------------------------------------------------
Total 14 cells                                            649818.000000
1
```

Power Report

```
Loading db file '/apps/design_kits/osu_stdcells_v2p7/synopsys/lib/ami05/osu05_stdcells.db'
Information: Propagating switching activity (low effort zero delay simulation). (PWR-6)
Warning: Design has unannotated primary inputs. (PWR-414)
Warning: Design has unannotated sequential cell outputs. (PWR-415)
```

```
****************************************
Report : power
        -analysis_effort low
Design : daq_top
Version: O-2018.06-SP1
Date   : Wed Dec 19 13:52:25 2018
****************************************


Library(s) Used:

    osu05_stdcells (File: /apps/design_kits/osu_stdcells_v2p7/synopsys/lib/ami05/osu05_stdcells.db)


Operating Conditions: typical   Library: osu05_stdcells
Wire Load Model Mode: top


Global Operating Voltage = 5
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000pf
    Time Units = 1ns
    Dynamic Power Units = 1mW    (derived from V,C,T units)
    Leakage Power Units = 1nW


  Cell Internal Power  =  21.2694 mW   (96%)
  Net Switching Power  = 786.2732 uW    (4%)
                         ---------
Total Dynamic Power    =  22.0557 mW  (100%)

Cell Leakage Power     = 209.5791 nW


                 Internal         Switching           Leakage            Total
Power Group      Power            Power               Power              Power   (   %    )  Attrs
```

```
--------------------------------------------------------------------------------------------------
io_pad              0.0000            0.0000            0.0000            0.0000  (   0.00%)
memory              0.0000            0.0000            0.0000            0.0000  (   0.00%)
black_box           0.0000            0.0000            0.0000            0.0000  (   0.00%)
clock_network       0.0000            0.0000            0.0000            0.0000  (   0.00%)
register           21.0231        8.8336e-02          107.1286           21.1116  (  95.72%)
sequential          0.0000            0.0000            0.0000            0.0000  (   0.00%)
combinational       0.2463            0.6979          102.4505            0.9443  (   4.28%)
--------------------------------------------------------------------------------------------------
Total              21.2694 mW         0.7863 mW       209.5791 nW        22.0559 mW
1
```

Appendix C

```
////////////////////////////////////////////////////////////
// Created by: Synopsys DC Expert(TM) in wire load mode
// Version   : O-2018.06-SP1
// Date      : Wed Dec 19 13:52:24 2018
////////////////////////////////////////////////////////////


module BCD_to_7seg_1 ( bcd, segs );
  input [11:0] bcd;
  output [20:0] segs;
  wire   n22, n23, n24, n25, n26, n27, n28, n29, n30, n31, n32, n33, n34, n35,
         n36, n37, n38, n39, n40, n41, n42, n43, n44, n45, n46, n47, n48, n49,
         n50, n51, n52, n53, n54, n55, n56, n57, n58, n59, n60, n1, n2, n3, n4,
         n5, n6, n7, n8, n9, n10, n11, n12, n13, n14, n15, n16, n17, n18;

  INVX2 U3 ( .A(n31), .Y(segs[3]) );
  INVX2 U4 ( .A(n42), .Y(segs[17]) );
  INVX2 U5 ( .A(n54), .Y(segs[10]) );
  OAI21X1 U24 ( .A(bcd[4]), .B(n22), .C(n10), .Y(segs[9]) );
  OAI21X1 U25 ( .A(n8), .B(n23), .C(n24), .Y(segs[8]) );
  AOI21X1 U26 ( .A(n25), .B(bcd[6]), .C(n26), .Y(n24) );
  NOR2X1 U27 ( .A(bcd[5]), .B(n6), .Y(n25) );
  NAND2X1 U28 ( .A(bcd[5]), .B(n6), .Y(n23) );
  OAI21X1 U29 ( .A(bcd[3]), .B(n3), .C(n2), .Y(segs[6]) );
  NAND2X1 U30 ( .A(n2), .B(n27), .Y(segs[5]) );
  NAND3X1 U31 ( .A(n29), .B(n4), .C(n30), .Y(n28) );
  NAND2X1 U32 ( .A(n31), .B(n1), .Y(segs[4]) );
  NOR3X1 U33 ( .A(n32), .B(segs[0]), .C(n33), .Y(n31) );
  OAI21X1 U34 ( .A(n1), .B(n3), .C(n30), .Y(n33) );
  NAND3X1 U35 ( .A(bcd[1]), .B(bcd[0]), .C(bcd[2]), .Y(n30) );
  OAI21X1 U36 ( .A(bcd[0]), .B(n27), .C(n4), .Y(segs[2]) );
  NAND2X1 U37 ( .A(bcd[1]), .B(n12), .Y(n27) );
  OAI21X1 U38 ( .A(bcd[11]), .B(n16), .C(n14), .Y(segs[20]) );
  OAI21X1 U39 ( .A(n12), .B(n35), .C(n36), .Y(segs[1]) );
  AOI21X1 U40 ( .A(n37), .B(bcd[2]), .C(n32), .Y(n36) );
  NOR2X1 U41 ( .A(n5), .B(n34), .Y(n32) );
```

```
NOR2X1 U42 ( .A(bcd[1]), .B(n1), .Y(n37) );
NAND2X1 U43 ( .A(bcd[1]), .B(n1), .Y(n35) );
NAND2X1 U44 ( .A(n14), .B(n38), .Y(segs[19]) );
NAND3X1 U45 ( .A(n40), .B(n17), .C(n41), .Y(n39) );
NAND2X1 U46 ( .A(n42), .B(n13), .Y(segs[18]) );
NOR3X1 U47 ( .A(n43), .B(segs[14]), .C(n44), .Y(n42) );
OAI21X1 U48 ( .A(n13), .B(n16), .C(n41), .Y(n44) );
NAND3X1 U49 ( .A(bcd[10]), .B(bcd[8]), .C(bcd[9]), .Y(n41) );
OAI21X1 U50 ( .A(bcd[8]), .B(n38), .C(n17), .Y(segs[16]) );
OR2X1 U51 ( .A(n15), .B(bcd[10]), .Y(n38) );
OAI21X1 U52 ( .A(n15), .B(n46), .C(n47), .Y(segs[15]) );
AOI21X1 U53 ( .A(n48), .B(bcd[10]), .C(n43), .Y(n47) );
NOR2X1 U54 ( .A(n18), .B(n45), .Y(n43) );
NOR2X1 U55 ( .A(bcd[9]), .B(n13), .Y(n48) );
NAND2X1 U56 ( .A(bcd[10]), .B(n13), .Y(n46) );
OAI21X1 U57 ( .A(n49), .B(n50), .C(n40), .Y(segs[14]) );
NAND3X1 U58 ( .A(n45), .B(n18), .C(bcd[8]), .Y(n40) );
NOR2X1 U59 ( .A(bcd[9]), .B(bcd[10]), .Y(n45) );
NAND2X1 U60 ( .A(bcd[10]), .B(n18), .Y(n50) );
NAND2X1 U61 ( .A(n13), .B(n15), .Y(n49) );
OAI21X1 U62 ( .A(bcd[7]), .B(n9), .C(n7), .Y(segs[13]) );
NAND2X1 U63 ( .A(n7), .B(n22), .Y(segs[12]) );
NAND2X1 U64 ( .A(bcd[5]), .B(n8), .Y(n22) );
NAND3X1 U65 ( .A(n52), .B(n10), .C(n53), .Y(n51) );
NAND2X1 U66 ( .A(n54), .B(n6), .Y(segs[11]) );
NOR3X1 U67 ( .A(n26), .B(segs[7]), .C(n55), .Y(n54) );
OAI21X1 U68 ( .A(n9), .B(n6), .C(n53), .Y(n55) );
NAND3X1 U69 ( .A(bcd[4]), .B(bcd[5]), .C(bcd[6]), .Y(n53) );
OAI21X1 U70 ( .A(n57), .B(n58), .C(n52), .Y(segs[7]) );
NAND3X1 U71 ( .A(bcd[4]), .B(n11), .C(n56), .Y(n52) );
NAND2X1 U72 ( .A(bcd[6]), .B(n6), .Y(n58) );
OR2X1 U73 ( .A(bcd[5]), .B(bcd[7]), .Y(n57) );
NOR2X1 U74 ( .A(n11), .B(n56), .Y(n26) );
NOR2X1 U75 ( .A(bcd[6]), .B(bcd[5]), .Y(n56) );
OAI21X1 U76 ( .A(n59), .B(n60), .C(n29), .Y(segs[0]) );
NAND3X1 U77 ( .A(n34), .B(n5), .C(bcd[0]), .Y(n29) );
NOR2X1 U78 ( .A(bcd[2]), .B(bcd[1]), .Y(n34) );
NAND2X1 U79 ( .A(bcd[2]), .B(n1), .Y(n60) );
```

```
  OR2X1 U80 ( .A(bcd[1]), .B(bcd[3]), .Y(n59) );
  INVX2 U6 ( .A(bcd[0]), .Y(n1) );
  INVX2 U7 ( .A(n28), .Y(n2) );
  INVX2 U8 ( .A(n34), .Y(n3) );
  INVX2 U9 ( .A(n32), .Y(n4) );
  INVX2 U10 ( .A(bcd[3]), .Y(n5) );
  INVX2 U11 ( .A(bcd[4]), .Y(n6) );
  INVX2 U12 ( .A(n51), .Y(n7) );
  INVX2 U13 ( .A(bcd[6]), .Y(n8) );
  INVX2 U14 ( .A(n56), .Y(n9) );
  INVX2 U15 ( .A(n26), .Y(n10) );
  INVX2 U16 ( .A(bcd[7]), .Y(n11) );
  INVX2 U17 ( .A(bcd[2]), .Y(n12) );
  INVX2 U18 ( .A(bcd[8]), .Y(n13) );
  INVX2 U19 ( .A(n39), .Y(n14) );
  INVX2 U20 ( .A(bcd[9]), .Y(n15) );
  INVX2 U21 ( .A(n45), .Y(n16) );
  INVX2 U22 ( .A(n43), .Y(n17) );
  INVX2 U23 ( .A(bcd[11]), .Y(n18) );
endmodule


module BCD_to_7seg_0 ( bcd, segs );
  input [11:0] bcd;
  output [20:0] segs;
  wire   n1, n2, n3, n4, n5, n6, n7, n8, n9, n10, n11, n12, n13, n14, n15, n16,
         n17, n18, n19, n20, n21, n58, n59, n60, n61, n62, n63, n64, n65, n66,
         n67, n68, n69, n70, n71, n72, n73, n74, n75, n76, n77, n78, n79, n80,
         n81, n82, n83, n84, n85, n86, n87, n88, n89, n90, n91, n92, n93;

  INVX2 U3 ( .A(n84), .Y(segs[3]) );
  INVX2 U4 ( .A(n73), .Y(segs[17]) );
  INVX2 U5 ( .A(n61), .Y(segs[10]) );
  OAI21X1 U24 ( .A(bcd[4]), .B(n93), .C(n9), .Y(segs[9]) );
  OAI21X1 U25 ( .A(n11), .B(n92), .C(n91), .Y(segs[8]) );
  AOI21X1 U26 ( .A(n90), .B(bcd[6]), .C(n89), .Y(n91) );
  NOR2X1 U27 ( .A(bcd[5]), .B(n6), .Y(n90) );
  NAND2X1 U28 ( .A(bcd[5]), .B(n6), .Y(n92) );
```

```
OAI21X1 U29 ( .A(bcd[3]), .B(n3), .C(n2), .Y(segs[6]) );
NAND2X1 U30 ( .A(n2), .B(n88), .Y(segs[5]) );
NAND3X1 U31 ( .A(n86), .B(n4), .C(n85), .Y(n87) );
NAND2X1 U32 ( .A(n84), .B(n1), .Y(segs[4]) );
NOR3X1 U33 ( .A(n83), .B(segs[0]), .C(n82), .Y(n84) );
OAI21X1 U34 ( .A(n1), .B(n3), .C(n85), .Y(n82) );
NAND3X1 U35 ( .A(bcd[1]), .B(bcd[0]), .C(bcd[2]), .Y(n85) );
OAI21X1 U36 ( .A(bcd[0]), .B(n88), .C(n4), .Y(segs[2]) );
NAND2X1 U37 ( .A(bcd[1]), .B(n12), .Y(n88) );
OAI21X1 U38 ( .A(bcd[11]), .B(n16), .C(n14), .Y(segs[20]) );
OAI21X1 U39 ( .A(n12), .B(n80), .C(n79), .Y(segs[1]) );
AOI21X1 U40 ( .A(n78), .B(bcd[2]), .C(n83), .Y(n79) );
NOR2X1 U41 ( .A(n5), .B(n81), .Y(n83) );
NOR2X1 U42 ( .A(bcd[1]), .B(n1), .Y(n78) );
NAND2X1 U43 ( .A(bcd[1]), .B(n1), .Y(n80) );
NAND2X1 U44 ( .A(n14), .B(n77), .Y(segs[19]) );
NAND3X1 U45 ( .A(n75), .B(n17), .C(n74), .Y(n76) );
NAND2X1 U46 ( .A(n73), .B(n13), .Y(segs[18]) );
NOR3X1 U47 ( .A(n72), .B(segs[14]), .C(n71), .Y(n73) );
OAI21X1 U48 ( .A(n13), .B(n16), .C(n74), .Y(n71) );
NAND3X1 U49 ( .A(bcd[10]), .B(bcd[8]), .C(bcd[9]), .Y(n74) );
OAI21X1 U50 ( .A(bcd[8]), .B(n77), .C(n17), .Y(segs[16]) );
OR2X1 U51 ( .A(n15), .B(bcd[10]), .Y(n77) );
OAI21X1 U52 ( .A(n15), .B(n69), .C(n68), .Y(segs[15]) );
AOI21X1 U53 ( .A(n67), .B(bcd[10]), .C(n72), .Y(n68) );
NOR2X1 U54 ( .A(n18), .B(n70), .Y(n72) );
NOR2X1 U55 ( .A(bcd[9]), .B(n13), .Y(n67) );
NAND2X1 U56 ( .A(bcd[10]), .B(n13), .Y(n69) );
OAI21X1 U57 ( .A(n66), .B(n65), .C(n75), .Y(segs[14]) );
NAND3X1 U58 ( .A(n70), .B(n18), .C(bcd[8]), .Y(n75) );
NOR2X1 U59 ( .A(bcd[9]), .B(bcd[10]), .Y(n70) );
NAND2X1 U60 ( .A(bcd[10]), .B(n18), .Y(n65) );
NAND2X1 U61 ( .A(n13), .B(n15), .Y(n66) );
OAI21X1 U62 ( .A(bcd[7]), .B(n8), .C(n7), .Y(segs[13]) );
NAND2X1 U63 ( .A(n7), .B(n93), .Y(segs[12]) );
NAND2X1 U64 ( .A(bcd[5]), .B(n11), .Y(n93) );
NAND3X1 U65 ( .A(n63), .B(n9), .C(n62), .Y(n64) );
NAND2X1 U66 ( .A(n61), .B(n6), .Y(segs[11]) );
```

```
  NOR3X1 U67 ( .A(n89), .B(segs[7]), .C(n60), .Y(n61) );
  OAI21X1 U68 ( .A(n8), .B(n6), .C(n62), .Y(n60) );
  NAND3X1 U69 ( .A(bcd[4]), .B(bcd[5]), .C(bcd[6]), .Y(n62) );
  OAI21X1 U70 ( .A(n58), .B(n21), .C(n63), .Y(segs[7]) );
  NAND3X1 U71 ( .A(bcd[4]), .B(n10), .C(n59), .Y(n63) );
  NAND2X1 U72 ( .A(bcd[6]), .B(n6), .Y(n21) );
  OR2X1 U73 ( .A(bcd[5]), .B(bcd[7]), .Y(n58) );
  NOR2X1 U74 ( .A(n10), .B(n59), .Y(n89) );
  NOR2X1 U75 ( .A(bcd[6]), .B(bcd[5]), .Y(n59) );
  OAI21X1 U76 ( .A(n20), .B(n19), .C(n86), .Y(segs[0]) );
  NAND3X1 U77 ( .A(n81), .B(n5), .C(bcd[0]), .Y(n86) );
  NOR2X1 U78 ( .A(bcd[2]), .B(bcd[1]), .Y(n81) );
  NAND2X1 U79 ( .A(bcd[2]), .B(n1), .Y(n19) );
  OR2X1 U80 ( .A(bcd[1]), .B(bcd[3]), .Y(n20) );
  INVX2 U6 ( .A(bcd[0]), .Y(n1) );
  INVX2 U7 ( .A(n87), .Y(n2) );
  INVX2 U8 ( .A(n81), .Y(n3) );
  INVX2 U9 ( .A(n83), .Y(n4) );
  INVX2 U10 ( .A(bcd[3]), .Y(n5) );
  INVX2 U11 ( .A(bcd[4]), .Y(n6) );
  INVX2 U12 ( .A(n64), .Y(n7) );
  INVX2 U13 ( .A(n59), .Y(n8) );
  INVX2 U14 ( .A(n89), .Y(n9) );
  INVX2 U15 ( .A(bcd[7]), .Y(n10) );
  INVX2 U16 ( .A(bcd[6]), .Y(n11) );
  INVX2 U17 ( .A(bcd[2]), .Y(n12) );
  INVX2 U18 ( .A(bcd[8]), .Y(n13) );
  INVX2 U19 ( .A(n76), .Y(n14) );
  INVX2 U20 ( .A(bcd[9]), .Y(n15) );
  INVX2 U21 ( .A(n70), .Y(n16) );
  INVX2 U22 ( .A(n72), .Y(n17) );
  INVX2 U23 ( .A(bcd[11]), .Y(n18) );
endmodule


module BCD_to_7seg_2 ( bcd, segs );
  input [7:0] bcd;
  output [13:0] segs;
```

```
wire   n14, n15, n16, n17, n18, n19, n20, n21, n22, n23, n24, n25, n26, n27,
       n28, n29, n30, n31, n32, n33, n34, n35, n36, n37, n38, n1, n2, n3, n4,
       n5, n6, n7, n8, n9, n10, n11, n12;

AND2X2 U3 ( .A(n19), .B(n20), .Y(segs[7]) );
INVX2 U4 ( .A(n25), .Y(segs[3]) );
OAI21X1 U17 ( .A(n8), .B(n14), .C(n15), .Y(segs[9]) );
NAND2X1 U18 ( .A(n7), .B(n11), .Y(n14) );
OAI21X1 U19 ( .A(n10), .B(n16), .C(n17), .Y(segs[8]) );
AOI21X1 U20 ( .A(n18), .B(bcd[6]), .C(n9), .Y(n17) );
NOR2X1 U21 ( .A(bcd[4]), .B(n8), .Y(n18) );
OAI21X1 U22 ( .A(bcd[3]), .B(n3), .C(n2), .Y(segs[6]) );
NAND2X1 U23 ( .A(n2), .B(n21), .Y(segs[5]) );
NAND3X1 U24 ( .A(n23), .B(n4), .C(n24), .Y(n22) );
NAND2X1 U25 ( .A(n25), .B(n1), .Y(segs[4]) );
NOR3X1 U26 ( .A(segs[0]), .B(n26), .C(n27), .Y(n25) );
OAI21X1 U27 ( .A(n1), .B(n3), .C(n24), .Y(n27) );
NAND3X1 U28 ( .A(bcd[1]), .B(bcd[0]), .C(bcd[2]), .Y(n24) );
OAI21X1 U29 ( .A(bcd[0]), .B(n21), .C(n4), .Y(segs[2]) );
NAND2X1 U30 ( .A(bcd[1]), .B(n12), .Y(n21) );
OAI21X1 U31 ( .A(n12), .B(n29), .C(n30), .Y(segs[1]) );
AOI21X1 U32 ( .A(n31), .B(bcd[2]), .C(n26), .Y(n30) );
NOR2X1 U33 ( .A(n5), .B(n28), .Y(n26) );
NOR2X1 U34 ( .A(bcd[1]), .B(n1), .Y(n31) );
NAND2X1 U35 ( .A(bcd[1]), .B(n1), .Y(n29) );
OAI21X1 U36 ( .A(bcd[6]), .B(n10), .C(n6), .Y(segs[13]) );
OAI21X1 U37 ( .A(bcd[6]), .B(n8), .C(n6), .Y(segs[12]) );
OR2X1 U38 ( .A(n32), .B(n33), .Y(segs[11]) );
OAI21X1 U39 ( .A(n11), .B(n10), .C(n7), .Y(n33) );
OAI21X1 U40 ( .A(n8), .B(n16), .C(n34), .Y(n32) );
AOI21X1 U41 ( .A(n35), .B(n20), .C(n9), .Y(n34) );
OAI21X1 U42 ( .A(bcd[5]), .B(bcd[6]), .C(bcd[7]), .Y(n15) );
NOR2X1 U43 ( .A(bcd[6]), .B(n7), .Y(n35) );
OAI21X1 U44 ( .A(n20), .B(n16), .C(n36), .Y(segs[10]) );
AOI22X1 U45 ( .A(n19), .B(n8), .C(bcd[7]), .D(bcd[5]), .Y(n36) );
XNOR2X1 U46 ( .A(bcd[4]), .B(n11), .Y(n19) );
NAND2X1 U47 ( .A(bcd[4]), .B(bcd[6]), .Y(n16) );
NOR2X1 U48 ( .A(bcd[7]), .B(bcd[5]), .Y(n20) );
```

```
  OAI21X1 U49 ( .A(n37), .B(n38), .C(n23), .Y(segs[0]) );
  NAND3X1 U50 ( .A(n28), .B(n5), .C(bcd[0]), .Y(n23) );
  NOR2X1 U51 ( .A(bcd[2]), .B(bcd[1]), .Y(n28) );
  NAND2X1 U52 ( .A(bcd[2]), .B(n1), .Y(n38) );
  OR2X1 U53 ( .A(bcd[1]), .B(bcd[3]), .Y(n37) );
  INVX2 U5 ( .A(bcd[0]), .Y(n1) );
  INVX2 U6 ( .A(n22), .Y(n2) );
  INVX2 U7 ( .A(n28), .Y(n3) );
  INVX2 U8 ( .A(n26), .Y(n4) );
  INVX2 U9 ( .A(bcd[3]), .Y(n5) );
  INVX2 U10 ( .A(n32), .Y(n6) );
  INVX2 U11 ( .A(bcd[4]), .Y(n7) );
  INVX2 U12 ( .A(bcd[5]), .Y(n8) );
  INVX2 U13 ( .A(n15), .Y(n9) );
  INVX2 U14 ( .A(n20), .Y(n10) );
  INVX2 U15 ( .A(bcd[6]), .Y(n11) );
  INVX2 U16 ( .A(bcd[2]), .Y(n12) );
endmodule


module serialOut_1_DW01_inc_0 ( A, SUM );
  input [29:0] A;
  output [29:0] SUM;

  wire   [29:2] carry;

  HAX1 U1_1_28 ( .A(A[28]), .B(carry[28]), .YC(carry[29]), .YS(SUM[28]) );
  HAX1 U1_1_27 ( .A(A[27]), .B(carry[27]), .YC(carry[28]), .YS(SUM[27]) );
  HAX1 U1_1_26 ( .A(A[26]), .B(carry[26]), .YC(carry[27]), .YS(SUM[26]) );
  HAX1 U1_1_25 ( .A(A[25]), .B(carry[25]), .YC(carry[26]), .YS(SUM[25]) );
  HAX1 U1_1_24 ( .A(A[24]), .B(carry[24]), .YC(carry[25]), .YS(SUM[24]) );
  HAX1 U1_1_23 ( .A(A[23]), .B(carry[23]), .YC(carry[24]), .YS(SUM[23]) );
  HAX1 U1_1_22 ( .A(A[22]), .B(carry[22]), .YC(carry[23]), .YS(SUM[22]) );
  HAX1 U1_1_21 ( .A(A[21]), .B(carry[21]), .YC(carry[22]), .YS(SUM[21]) );
  HAX1 U1_1_20 ( .A(A[20]), .B(carry[20]), .YC(carry[21]), .YS(SUM[20]) );
  HAX1 U1_1_19 ( .A(A[19]), .B(carry[19]), .YC(carry[20]), .YS(SUM[19]) );
  HAX1 U1_1_18 ( .A(A[18]), .B(carry[18]), .YC(carry[19]), .YS(SUM[18]) );
  HAX1 U1_1_17 ( .A(A[17]), .B(carry[17]), .YC(carry[18]), .YS(SUM[17]) );
```

```
  HAX1 U1_1_16 ( .A(A[16]), .B(carry[16]), .YC(carry[17]), .YS(SUM[16]) );
  HAX1 U1_1_15 ( .A(A[15]), .B(carry[15]), .YC(carry[16]), .YS(SUM[15]) );
  HAX1 U1_1_14 ( .A(A[14]), .B(carry[14]), .YC(carry[15]), .YS(SUM[14]) );
  HAX1 U1_1_13 ( .A(A[13]), .B(carry[13]), .YC(carry[14]), .YS(SUM[13]) );
  HAX1 U1_1_12 ( .A(A[12]), .B(carry[12]), .YC(carry[13]), .YS(SUM[12]) );
  HAX1 U1_1_11 ( .A(A[11]), .B(carry[11]), .YC(carry[12]), .YS(SUM[11]) );
  HAX1 U1_1_10 ( .A(A[10]), .B(carry[10]), .YC(carry[11]), .YS(SUM[10]) );
  HAX1 U1_1_9 ( .A(A[9]), .B(carry[9]), .YC(carry[10]), .YS(SUM[9]) );
  HAX1 U1_1_8 ( .A(A[8]), .B(carry[8]), .YC(carry[9]), .YS(SUM[8]) );
  HAX1 U1_1_7 ( .A(A[7]), .B(carry[7]), .YC(carry[8]), .YS(SUM[7]) );
  HAX1 U1_1_6 ( .A(A[6]), .B(carry[6]), .YC(carry[7]), .YS(SUM[6]) );
  HAX1 U1_1_5 ( .A(A[5]), .B(carry[5]), .YC(carry[6]), .YS(SUM[5]) );
  HAX1 U1_1_4 ( .A(A[4]), .B(carry[4]), .YC(carry[5]), .YS(SUM[4]) );
  HAX1 U1_1_3 ( .A(A[3]), .B(carry[3]), .YC(carry[4]), .YS(SUM[3]) );
  HAX1 U1_1_2 ( .A(A[2]), .B(carry[2]), .YC(carry[3]), .YS(SUM[2]) );
  HAX1 U1_1_1 ( .A(A[1]), .B(A[0]), .YC(carry[2]), .YS(SUM[1]) );
  XOR2X1 U1 ( .A(carry[29]), .B(A[29]), .Y(SUM[29]) );
  INVX1 U2 ( .A(A[0]), .Y(SUM[0]) );
endmodule


module serialOut_1 ( clk, rst, initiate, data_in, data_out, trx_busy );
  input [9:0] data_in;
  input clk, rst, initiate;
  output data_out, trx_busy;
  wire   trx_finished_count, transmit, initiate_dly, N44, N45, N46, N47, N48,
         N49, N50, N51, N52, N53, N54, N55, N56, N57, N58, N59, N60, N61, N62,
         N63, N64, N65, N66, N67, N68, N69, N70, N71, N72, N73, N601, N602,
         n198, n199, n1, n2, n3, n4, n5, n6, n7, n8, n9, n10, n11, n12, n13,
         n14, n15, n16, n17, n18, n19, n20, n21, n22, n23, n24, n25, n26, n27,
         n28, n29, n30, n31, n32, n67, n69, n70, n71, n72, n73, n74, n75, n76,
         n77, n78, n79, n80, n81, n82, n83, n84, n85, n86, n87, n88, n89, n90,
         n91, n92, n93, n94, n95, n96, n97, n98, n99, n100, n101, n102, n103,
         n104, n105, n106, n107, n108, n109, n110, n111, n112, n113, n114,
         n115, n116, n117, n118, n119, n120, n121, n122, n123, n124, n125,
         n126, n127, n128, n129, n130, n131, n132, n133, n134, n135, n136,
         n137, n138, n139, n140, n141, n142, n143, n144, n145, n146, n147,
         n148, n149, n150, n151, n152, n153, n154, n155, n156, n157, n158,
```

```
        n159, n160, n161, n162, n163, n164, n165, n166, n167, n168, n169,
        n170, n171, n172, n173, n174, n175, n176, n177, n178, n179, n180,
        n181, n182, n183, n184, n185, n186, n187, n188, n189, n190, n191,
        n192, n193, n194, n195, n196, n197, n200, n201, n202, n203, n204,
        n205, n206, n207, n208, n209, n210, n211, n212, n213, n214, n215,
        n216, n217, n218, n219, n220, n221, n222, n223, n224, n225, n226,
        n227, n228, n229, n230, n231, n232, n233, n234, n235, n236, n237,
        n238, n239, n240, n241, n242, n243, n244, n245, n246, n247, n248,
        n249, n250, n251, n252, n253, n254, n255, n256, n257, n258, n259,
        n260, n261, n262, n263, n264, n265, n266, n267, n268, n269, n270,
        n271, n272, n273, n274, n275;
wire   [29:0] trx_delay_counter;

DFFSR initiate_dly_reg ( .D(initiate), .CLK(clk), .R(n6), .S(1'b1), .Q(
      initiate_dly) );
DFFSR trx_busy_reg ( .D(n198), .CLK(clk), .R(n6), .S(1'b1), .Q(trx_busy) );
DFFSR transmit_reg ( .D(n199), .CLK(clk), .R(n7), .S(1'b1), .Q(transmit) );
DFFSR trx_delay_counter_reg_29_ ( .D(n246), .CLK(clk), .R(n6), .S(1'b1), .Q(
      trx_delay_counter[29]) );
DFFSR trx_delay_counter_reg_28_ ( .D(n247), .CLK(clk), .R(n7), .S(1'b1), .Q(
      trx_delay_counter[28]) );
DFFSR trx_delay_counter_reg_27_ ( .D(n248), .CLK(clk), .R(n6), .S(1'b1), .Q(
      trx_delay_counter[27]) );
DFFSR trx_delay_counter_reg_26_ ( .D(n249), .CLK(clk), .R(n7), .S(1'b1), .Q(
      trx_delay_counter[26]) );
DFFSR trx_delay_counter_reg_25_ ( .D(n250), .CLK(clk), .R(n6), .S(1'b1), .Q(
      trx_delay_counter[25]) );
DFFSR trx_delay_counter_reg_24_ ( .D(n251), .CLK(clk), .R(n7), .S(1'b1), .Q(
      trx_delay_counter[24]) );
DFFSR trx_delay_counter_reg_23_ ( .D(n252), .CLK(clk), .R(n6), .S(1'b1), .Q(
      trx_delay_counter[23]) );
DFFSR trx_delay_counter_reg_22_ ( .D(n253), .CLK(clk), .R(n7), .S(1'b1), .Q(
      trx_delay_counter[22]) );
DFFSR trx_delay_counter_reg_21_ ( .D(n254), .CLK(clk), .R(n7), .S(1'b1), .Q(
      trx_delay_counter[21]) );
DFFSR trx_delay_counter_reg_20_ ( .D(n255), .CLK(clk), .R(n7), .S(1'b1), .Q(
      trx_delay_counter[20]) );
DFFSR trx_delay_counter_reg_19_ ( .D(n256), .CLK(clk), .R(n7), .S(1'b1), .Q(
```

```
      trx_delay_counter[19]) );
DFFSR trx_delay_counter_reg_18_ ( .D(n257), .CLK(clk), .R(n7), .S(1'b1), .Q(
      trx_delay_counter[18]) );
DFFSR trx_delay_counter_reg_17_ ( .D(n258), .CLK(clk), .R(n7), .S(1'b1), .Q(
      trx_delay_counter[17]) );
DFFSR trx_delay_counter_reg_16_ ( .D(n259), .CLK(clk), .R(n7), .S(1'b1), .Q(
      trx_delay_counter[16]) );
DFFSR trx_delay_counter_reg_15_ ( .D(n260), .CLK(clk), .R(n7), .S(1'b1), .Q(
      trx_delay_counter[15]) );
DFFSR trx_delay_counter_reg_14_ ( .D(n261), .CLK(clk), .R(n7), .S(1'b1), .Q(
      trx_delay_counter[14]) );
DFFSR trx_delay_counter_reg_13_ ( .D(n262), .CLK(clk), .R(n7), .S(1'b1), .Q(
      trx_delay_counter[13]) );
DFFSR trx_delay_counter_reg_12_ ( .D(n263), .CLK(clk), .R(n7), .S(1'b1), .Q(
      trx_delay_counter[12]) );
DFFSR trx_delay_counter_reg_11_ ( .D(n264), .CLK(clk), .R(n7), .S(1'b1), .Q(
      trx_delay_counter[11]) );
DFFSR trx_delay_counter_reg_10_ ( .D(n265), .CLK(clk), .R(n7), .S(1'b1), .Q(
      trx_delay_counter[10]) );
DFFSR trx_delay_counter_reg_9_ ( .D(n266), .CLK(clk), .R(n6), .S(1'b1), .Q(
      trx_delay_counter[9]) );
DFFSR trx_delay_counter_reg_8_ ( .D(n267), .CLK(clk), .R(n6), .S(1'b1), .Q(
      trx_delay_counter[8]) );
DFFSR trx_delay_counter_reg_7_ ( .D(n268), .CLK(clk), .R(n6), .S(1'b1), .Q(
      trx_delay_counter[7]) );
DFFSR trx_delay_counter_reg_6_ ( .D(n269), .CLK(clk), .R(n6), .S(1'b1), .Q(
      trx_delay_counter[6]) );
DFFSR trx_delay_counter_reg_5_ ( .D(n270), .CLK(clk), .R(n6), .S(1'b1), .Q(
      trx_delay_counter[5]) );
DFFSR trx_delay_counter_reg_4_ ( .D(n271), .CLK(clk), .R(n6), .S(1'b1), .Q(
      trx_delay_counter[4]) );
DFFSR trx_delay_counter_reg_3_ ( .D(n272), .CLK(clk), .R(n6), .S(1'b1), .Q(
      trx_delay_counter[3]) );
DFFSR trx_delay_counter_reg_2_ ( .D(n273), .CLK(clk), .R(n6), .S(1'b1), .Q(
      trx_delay_counter[2]) );
DFFSR trx_delay_counter_reg_1_ ( .D(n274), .CLK(clk), .R(n6), .S(1'b1), .Q(
      trx_delay_counter[1]) );
DFFSR trx_delay_counter_reg_0_ ( .D(n275), .CLK(clk), .R(n6), .S(1'b1), .Q(
```

```
      trx_delay_counter[0]) );
DFFSR trx_finished_count_reg ( .D(N601), .CLK(clk), .R(n6), .S(1'b1), .Q(
      trx_finished_count) );
DFFSR data_out_reg ( .D(N602), .CLK(clk), .R(n6), .S(1'b1), .Q(data_out) );
serialOut_1_DW01_inc_0 r136 ( .A({trx_delay_counter[29:25], n5,
      trx_delay_counter[23:21], n4, trx_delay_counter[19:15], n3, n2,
      trx_delay_counter[12:0]}), .SUM({N73, N72, N71, N70, N69, N68, N67,
      N66, N65, N64, N63, N62, N61, N60, N59, N58, N57, N56, N55, N54, N53,
      N52, N51, N50, N49, N48, N47, N46, N45, N44}) );
AND2X2 U5 ( .A(transmit), .B(n8), .Y(n1) );
INVX2 U6 ( .A(rst), .Y(n6) );
INVX2 U7 ( .A(rst), .Y(n7) );
BUFX2 U8 ( .A(trx_delay_counter[14]), .Y(n3) );
BUFX2 U9 ( .A(trx_delay_counter[13]), .Y(n2) );
BUFX2 U10 ( .A(trx_delay_counter[20]), .Y(n4) );
BUFX2 U11 ( .A(trx_delay_counter[24]), .Y(n5) );
AND2X1 U12 ( .A(N73), .B(n1), .Y(n246) );
AND2X1 U13 ( .A(N72), .B(n1), .Y(n247) );
AND2X1 U14 ( .A(N71), .B(n1), .Y(n248) );
AND2X1 U15 ( .A(N70), .B(n1), .Y(n249) );
AND2X1 U16 ( .A(N69), .B(n1), .Y(n250) );
AND2X1 U17 ( .A(N68), .B(n1), .Y(n251) );
AND2X1 U18 ( .A(N67), .B(n1), .Y(n252) );
AND2X1 U19 ( .A(N66), .B(n1), .Y(n253) );
AND2X1 U20 ( .A(N65), .B(n1), .Y(n254) );
AND2X1 U53 ( .A(N64), .B(n1), .Y(n255) );
AND2X1 U55 ( .A(N63), .B(n1), .Y(n256) );
AND2X1 U56 ( .A(N62), .B(n1), .Y(n257) );
AND2X1 U57 ( .A(N61), .B(n1), .Y(n258) );
AND2X1 U58 ( .A(N60), .B(n1), .Y(n259) );
AND2X1 U59 ( .A(N59), .B(n1), .Y(n260) );
AND2X1 U60 ( .A(N58), .B(n1), .Y(n261) );
AND2X1 U61 ( .A(N57), .B(n1), .Y(n262) );
AND2X1 U62 ( .A(N56), .B(n1), .Y(n263) );
AND2X1 U63 ( .A(N55), .B(n1), .Y(n264) );
AND2X1 U64 ( .A(N54), .B(n1), .Y(n265) );
AND2X1 U65 ( .A(N53), .B(n1), .Y(n266) );
AND2X1 U66 ( .A(N52), .B(n1), .Y(n267) );
```

```
AND2X1 U67 ( .A(N51), .B(n1), .Y(n268) );
AND2X1 U68 ( .A(N50), .B(n1), .Y(n269) );
AND2X1 U69 ( .A(N49), .B(n1), .Y(n270) );
AND2X1 U70 ( .A(N48), .B(n1), .Y(n271) );
AND2X1 U71 ( .A(N47), .B(n1), .Y(n272) );
AND2X1 U72 ( .A(N46), .B(n1), .Y(n273) );
AND2X1 U73 ( .A(N45), .B(n1), .Y(n274) );
AND2X1 U74 ( .A(N44), .B(n1), .Y(n275) );
MUX2X1 U75 ( .B(trx_finished_count), .A(n9), .S(n10), .Y(n199) );
MUX2X1 U76 ( .B(trx_finished_count), .A(n11), .S(n10), .Y(n198) );
AOI21X1 U77 ( .A(n12), .B(initiate), .C(trx_finished_count), .Y(n10) );
INVX1 U78 ( .A(initiate_dly), .Y(n12) );
INVX1 U79 ( .A(trx_busy), .Y(n11) );
AOI21X1 U80 ( .A(n13), .B(n14), .C(n9), .Y(N602) );
NOR2X1 U81 ( .A(n15), .B(n16), .Y(n14) );
OAI22X1 U82 ( .A(n17), .B(n18), .C(n19), .D(n20), .Y(n16) );
MUX2X1 U83 ( .B(n21), .A(data_in[7]), .S(n22), .Y(n20) );
OAI21X1 U84 ( .A(n23), .B(n24), .C(n25), .Y(n22) );
OAI21X1 U85 ( .A(n26), .B(n27), .C(n28), .Y(n24) );
NOR2X1 U86 ( .A(n29), .B(n30), .Y(n26) );
OAI21X1 U87 ( .A(n31), .B(n32), .C(n67), .Y(n30) );
OAI21X1 U88 ( .A(n69), .B(n70), .C(trx_delay_counter[16]), .Y(n32) );
OAI21X1 U89 ( .A(n71), .B(n72), .C(n73), .Y(n70) );
INVX1 U90 ( .A(n74), .Y(n69) );
OAI21X1 U91 ( .A(trx_delay_counter[10]), .B(n75), .C(n2), .Y(n74) );
NAND2X1 U92 ( .A(n76), .B(n77), .Y(n29) );
AND2X1 U93 ( .A(data_in[6]), .B(n78), .Y(n21) );
NAND2X1 U94 ( .A(data_in[0]), .B(n79), .Y(n18) );
INVX1 U95 ( .A(n80), .Y(n79) );
OAI21X1 U96 ( .A(n81), .B(n82), .C(n83), .Y(n17) );
NOR2X1 U97 ( .A(trx_delay_counter[28]), .B(trx_delay_counter[27]), .Y(n83)
      );
NAND2X1 U98 ( .A(trx_delay_counter[26]), .B(n84), .Y(n82) );
OAI21X1 U99 ( .A(n85), .B(n86), .C(n87), .Y(n84) );
NOR2X1 U100 ( .A(n5), .B(n88), .Y(n87) );
NAND3X1 U101 ( .A(n4), .B(trx_delay_counter[21]), .C(n89), .Y(n86) );
OR2X1 U102 ( .A(n90), .B(n91), .Y(n85) );
OAI21X1 U103 ( .A(n92), .B(n93), .C(trx_delay_counter[23]), .Y(n91) );
```

```
OAI21X1 U104 ( .A(n94), .B(n95), .C(n73), .Y(n93) );
OAI21X1 U105 ( .A(n96), .B(trx_delay_counter[11]), .C(n2), .Y(n95) );
AND2X1 U106 ( .A(trx_delay_counter[10]), .B(n97), .Y(n96) );
MUX2X1 U107 ( .B(n98), .A(n99), .S(n100), .Y(n15) );
NOR2X1 U108 ( .A(trx_delay_counter[27]), .B(n25), .Y(n100) );
AOI21X1 U109 ( .A(n81), .B(n101), .C(n102), .Y(n99) );
NAND3X1 U110 ( .A(n103), .B(n104), .C(n4), .Y(n101) );
OAI21X1 U111 ( .A(n105), .B(n106), .C(n107), .Y(n104) );
NOR2X1 U112 ( .A(trx_delay_counter[19]), .B(n89), .Y(n107) );
NAND3X1 U113 ( .A(n3), .B(n2), .C(trx_delay_counter[15]), .Y(n106) );
NAND3X1 U114 ( .A(trx_delay_counter[16]), .B(n108), .C(trx_delay_counter[18]), .Y(n105) );
NAND3X1 U115 ( .A(n71), .B(n109), .C(n110), .Y(n108) );
OAI21X1 U116 ( .A(n111), .B(n112), .C(n113), .Y(n98) );
MUX2X1 U117 ( .B(n114), .A(n115), .S(n116), .Y(n113) );
NOR2X1 U118 ( .A(n25), .B(n117), .Y(n116) );
OAI21X1 U119 ( .A(n27), .B(n118), .C(n102), .Y(n117) );
OAI21X1 U120 ( .A(n119), .B(n4), .C(trx_delay_counter[25]), .Y(n118) );
AOI21X1 U121 ( .A(n120), .B(n121), .C(n90), .Y(n119) );
NAND3X1 U122 ( .A(trx_delay_counter[15]), .B(n3), .C(n122), .Y(n121) );
AOI21X1 U123 ( .A(n123), .B(n110), .C(n124), .Y(n122) );
NOR2X1 U124 ( .A(trx_delay_counter[10]), .B(trx_delay_counter[9]), .Y(n110)
      );
NOR2X1 U125 ( .A(n2), .B(n125), .Y(n123) );
INVX1 U126 ( .A(n103), .Y(n27) );
INVX1 U127 ( .A(data_in[9]), .Y(n115) );
NAND2X1 U128 ( .A(data_in[8]), .B(n19), .Y(n114) );
INVX1 U129 ( .A(n126), .Y(n19) );
OAI21X1 U130 ( .A(n81), .B(n127), .C(n128), .Y(n126) );
OAI21X1 U131 ( .A(n129), .B(n5), .C(n130), .Y(n127) );
AOI21X1 U132 ( .A(n131), .B(n132), .C(n133), .Y(n129) );
NAND3X1 U133 ( .A(n134), .B(n92), .C(n89), .Y(n132) );
OAI21X1 U134 ( .A(n94), .B(n135), .C(n136), .Y(n134) );
NOR2X1 U135 ( .A(trx_delay_counter[16]), .B(n3), .Y(n136) );
OAI21X1 U136 ( .A(n97), .B(n137), .C(n2), .Y(n135) );
OR2X1 U137 ( .A(trx_delay_counter[10]), .B(trx_delay_counter[11]), .Y(n137)
      );
NOR2X1 U138 ( .A(n138), .B(n139), .Y(n97) );
NOR2X1 U139 ( .A(n4), .B(trx_delay_counter[19]), .Y(n131) );
```

```
OAI21X1 U140 ( .A(n77), .B(n140), .C(n141), .Y(n112) );
OAI21X1 U141 ( .A(n142), .B(n143), .C(n144), .Y(n140) );
OAI21X1 U142 ( .A(n31), .B(n145), .C(n67), .Y(n143) );
OAI21X1 U143 ( .A(n146), .B(n147), .C(n3), .Y(n145) );
AND2X1 U144 ( .A(n148), .B(trx_delay_counter[12]), .Y(n146) );
INVX1 U145 ( .A(n28), .Y(n111) );
NOR2X1 U146 ( .A(n149), .B(n150), .Y(n13) );
MUX2X1 U147 ( .B(n151), .A(n152), .S(n153), .Y(n150) );
NAND3X1 U148 ( .A(n154), .B(n155), .C(data_in[5]), .Y(n152) );
INVX1 U149 ( .A(n156), .Y(n154) );
NAND3X1 U150 ( .A(n8), .B(n157), .C(n130), .Y(n151) );
OAI21X1 U151 ( .A(n158), .B(n159), .C(n160), .Y(n157) );
OAI21X1 U152 ( .A(trx_delay_counter[22]), .B(n4), .C(n161), .Y(n159) );
NAND3X1 U153 ( .A(n90), .B(n162), .C(n163), .Y(n161) );
OAI21X1 U154 ( .A(n164), .B(n165), .C(trx_delay_counter[18]), .Y(n163) );
NAND2X1 U155 ( .A(n166), .B(n167), .Y(n165) );
OAI21X1 U156 ( .A(n168), .B(n169), .C(n170), .Y(n164) );
NOR2X1 U157 ( .A(trx_delay_counter[17]), .B(n3), .Y(n170) );
INVX1 U158 ( .A(trx_delay_counter[22]), .Y(n162) );
MUX2X1 U159 ( .B(n171), .A(n172), .S(n173), .Y(n149) );
AOI21X1 U160 ( .A(n174), .B(n153), .C(n128), .Y(n173) );
NAND2X1 U161 ( .A(trx_delay_counter[27]), .B(n175), .Y(n174) );
OAI21X1 U162 ( .A(n176), .B(n81), .C(n102), .Y(n175) );
AOI21X1 U163 ( .A(n88), .B(n177), .C(n5), .Y(n176) );
OAI21X1 U164 ( .A(n178), .B(n77), .C(n179), .Y(n177) );
AOI21X1 U165 ( .A(trx_delay_counter[17]), .B(n180), .C(n181), .Y(n178) );
INVX1 U166 ( .A(n67), .Y(n181) );
OAI21X1 U167 ( .A(n182), .B(n183), .C(n166), .Y(n180) );
NAND2X1 U168 ( .A(trx_delay_counter[12]), .B(n3), .Y(n183) );
OAI21X1 U169 ( .A(trx_delay_counter[11]), .B(n184), .C(n2), .Y(n182) );
AOI21X1 U170 ( .A(n139), .B(n138), .C(n168), .Y(n184) );
NOR2X1 U171 ( .A(trx_delay_counter[8]), .B(trx_delay_counter[7]), .Y(n139)
      );
MUX2X1 U172 ( .B(n185), .A(data_in[3]), .S(n186), .Y(n172) );
AOI21X1 U173 ( .A(n130), .B(n187), .C(trx_delay_counter[29]), .Y(n186) );
OAI21X1 U174 ( .A(n188), .B(n189), .C(n81), .Y(n187) );
NAND2X1 U175 ( .A(n5), .B(n190), .Y(n189) );
OAI21X1 U176 ( .A(n90), .B(n191), .C(n192), .Y(n190) );
```

```
NOR2X1 U177 ( .A(trx_delay_counter[21]), .B(n4), .Y(n192) );
OAI21X1 U178 ( .A(trx_delay_counter[17]), .B(n193), .C(trx_delay_counter[18]), .Y(n191) );
AND2X1 U179 ( .A(trx_delay_counter[16]), .B(n194), .Y(n193) );
OAI21X1 U180 ( .A(n73), .B(n195), .C(n31), .Y(n194) );
OAI21X1 U181 ( .A(n196), .B(n125), .C(n2), .Y(n195) );
AOI21X1 U182 ( .A(n109), .B(n138), .C(n168), .Y(n196) );
MUX2X1 U183 ( .B(n197), .A(n200), .S(n201), .Y(n185) );
OAI21X1 U184 ( .A(n202), .B(n203), .C(trx_delay_counter[29]), .Y(n201) );
OAI21X1 U185 ( .A(n204), .B(n188), .C(n28), .Y(n203) );
NOR2X1 U186 ( .A(trx_delay_counter[26]), .B(trx_delay_counter[25]), .Y(n28)
      );
AOI21X1 U187 ( .A(trx_delay_counter[19]), .B(n205), .C(n206), .Y(n204) );
NAND3X1 U188 ( .A(n76), .B(n120), .C(n207), .Y(n205) );
AOI21X1 U189 ( .A(n3), .B(n208), .C(n92), .Y(n207) );
INVX1 U190 ( .A(n166), .Y(n92) );
NOR2X1 U191 ( .A(trx_delay_counter[16]), .B(trx_delay_counter[15]), .Y(n166)
      );
OAI21X1 U192 ( .A(n168), .B(n209), .C(n167), .Y(n208) );
NAND2X1 U193 ( .A(trx_delay_counter[12]), .B(n148), .Y(n209) );
OAI21X1 U194 ( .A(n210), .B(n109), .C(n138), .Y(n148) );
INVX1 U195 ( .A(trx_delay_counter[7]), .Y(n210) );
NAND3X1 U196 ( .A(n211), .B(n212), .C(n141), .Y(n202) );
INVX1 U197 ( .A(data_in[2]), .Y(n200) );
NAND2X1 U198 ( .A(data_in[1]), .B(n80), .Y(n197) );
OAI21X1 U199 ( .A(n213), .B(n214), .C(trx_delay_counter[29]), .Y(n80) );
OAI21X1 U200 ( .A(n215), .B(n216), .C(n102), .Y(n214) );
NAND2X1 U201 ( .A(n88), .B(trx_delay_counter[25]), .Y(n216) );
INVX1 U202 ( .A(n188), .Y(n88) );
OAI21X1 U203 ( .A(n206), .B(n217), .C(n5), .Y(n215) );
OAI21X1 U204 ( .A(n218), .B(n124), .C(n67), .Y(n217) );
NOR2X1 U205 ( .A(trx_delay_counter[19]), .B(trx_delay_counter[18]), .Y(n67)
      );
INVX1 U206 ( .A(n219), .Y(n124) );
AND2X1 U207 ( .A(n31), .B(n220), .Y(n218) );
OAI21X1 U208 ( .A(n125), .B(n221), .C(n3), .Y(n220) );
OAI21X1 U209 ( .A(n168), .B(n138), .C(n72), .Y(n221) );
INVX1 U210 ( .A(n71), .Y(n125) );
NAND2X1 U211 ( .A(n77), .B(n179), .Y(n206) );
```

```
NAND2X1 U212 ( .A(n211), .B(n212), .Y(n213) );
NAND3X1 U213 ( .A(n155), .B(n156), .C(data_in[4]), .Y(n171) );
OAI21X1 U214 ( .A(n222), .B(n81), .C(n211), .Y(n156) );
INVX1 U215 ( .A(trx_delay_counter[25]), .Y(n81) );
AOI21X1 U216 ( .A(n223), .B(n224), .C(n103), .Y(n222) );
NOR2X1 U217 ( .A(n133), .B(n141), .Y(n103) );
INVX1 U218 ( .A(n144), .Y(n133) );
NOR2X1 U219 ( .A(n179), .B(n188), .Y(n144) );
AOI21X1 U220 ( .A(n90), .B(n225), .C(n141), .Y(n224) );
INVX1 U221 ( .A(n5), .Y(n141) );
NAND3X1 U222 ( .A(n89), .B(trx_delay_counter[15]), .C(n226), .Y(n225) );
INVX1 U223 ( .A(n227), .Y(n226) );
OAI21X1 U224 ( .A(n228), .B(trx_delay_counter[10]), .C(trx_delay_counter[16]), .Y(n227) );
NOR2X1 U225 ( .A(n76), .B(n120), .Y(n89) );
INVX1 U226 ( .A(trx_delay_counter[18]), .Y(n120) );
INVX1 U227 ( .A(trx_delay_counter[19]), .Y(n90) );
NOR2X1 U228 ( .A(n188), .B(n77), .Y(n223) );
INVX1 U229 ( .A(n4), .Y(n77) );
NAND2X1 U230 ( .A(trx_delay_counter[22]), .B(trx_delay_counter[23]), .Y(n188) );
INVX1 U231 ( .A(n78), .Y(n155) );
OAI21X1 U232 ( .A(n229), .B(n23), .C(n25), .Y(n78) );
INVX1 U233 ( .A(n128), .Y(n25) );
NOR2X1 U234 ( .A(trx_delay_counter[28]), .B(trx_delay_counter[29]), .Y(n128)
      );
NAND2X1 U235 ( .A(n211), .B(n153), .Y(n23) );
INVX1 U236 ( .A(trx_delay_counter[29]), .Y(n153) );
AOI21X1 U237 ( .A(n160), .B(n230), .C(n102), .Y(n229) );
NAND3X1 U238 ( .A(trx_delay_counter[23]), .B(n231), .C(trx_delay_counter[22]), .Y(n230) );
OAI21X1 U239 ( .A(n232), .B(n233), .C(n179), .Y(n231) );
INVX1 U240 ( .A(trx_delay_counter[21]), .Y(n179) );
NAND2X1 U241 ( .A(n4), .B(trx_delay_counter[19]), .Y(n233) );
OAI21X1 U242 ( .A(n234), .B(n142), .C(trx_delay_counter[18]), .Y(n232) );
NAND2X1 U243 ( .A(n235), .B(n76), .Y(n142) );
AOI21X1 U244 ( .A(n236), .B(n237), .C(n31), .Y(n234) );
INVX1 U245 ( .A(trx_delay_counter[15]), .Y(n31) );
INVX1 U246 ( .A(n147), .Y(n237) );
OAI21X1 U247 ( .A(n94), .B(n168), .C(n167), .Y(n147) );
AOI21X1 U248 ( .A(trx_delay_counter[11]), .B(trx_delay_counter[12]), .C(n2),
```

```
      .Y(n167) );
  INVX1 U249 ( .A(trx_delay_counter[10]), .Y(n168) );
  INVX1 U250 ( .A(trx_delay_counter[12]), .Y(n94) );
  AND2X1 U251 ( .A(n73), .B(n169), .Y(n236) );
  NAND3X1 U252 ( .A(trx_delay_counter[7]), .B(trx_delay_counter[12]), .C(n75),
      .Y(n169) );
  NOR2X1 U253 ( .A(n138), .B(n109), .Y(n75) );
  INVX1 U254 ( .A(trx_delay_counter[8]), .Y(n109) );
  INVX1 U255 ( .A(trx_delay_counter[9]), .Y(n138) );
  NOR2X1 U256 ( .A(trx_delay_counter[25]), .B(n5), .Y(n160) );
  NOR2X1 U257 ( .A(n9), .B(n8), .Y(N601) );
  NAND2X1 U258 ( .A(trx_delay_counter[29]), .B(n238), .Y(n8) );
  OAI21X1 U259 ( .A(n239), .B(n240), .C(n212), .Y(n238) );
  INVX1 U260 ( .A(trx_delay_counter[28]), .Y(n212) );
  NAND2X1 U261 ( .A(trx_delay_counter[25]), .B(n5), .Y(n240) );
  NAND3X1 U262 ( .A(n241), .B(n242), .C(n130), .Y(n239) );
  NOR2X1 U263 ( .A(n211), .B(n102), .Y(n130) );
  INVX1 U264 ( .A(trx_delay_counter[26]), .Y(n102) );
  INVX1 U265 ( .A(trx_delay_counter[27]), .Y(n211) );
  OAI21X1 U266 ( .A(n243), .B(n244), .C(n245), .Y(n242) );
  NOR2X1 U267 ( .A(trx_delay_counter[22]), .B(n4), .Y(n245) );
  NAND2X1 U268 ( .A(trx_delay_counter[19]), .B(trx_delay_counter[18]), .Y(n244) );
  OAI21X1 U269 ( .A(trx_delay_counter[15]), .B(n228), .C(n219), .Y(n243) );
  NOR2X1 U270 ( .A(n76), .B(n235), .Y(n219) );
  INVX1 U271 ( .A(trx_delay_counter[16]), .Y(n235) );
  INVX1 U272 ( .A(trx_delay_counter[17]), .Y(n76) );
  NAND3X1 U273 ( .A(n72), .B(n73), .C(n71), .Y(n228) );
  NOR2X1 U274 ( .A(trx_delay_counter[12]), .B(trx_delay_counter[11]), .Y(n71)
       );
  INVX1 U275 ( .A(n3), .Y(n73) );
  INVX1 U276 ( .A(n2), .Y(n72) );
  INVX1 U277 ( .A(n158), .Y(n241) );
  OAI21X1 U278 ( .A(trx_delay_counter[22]), .B(trx_delay_counter[21]), .C(
      trx_delay_counter[23]), .Y(n158) );
  INVX1 U279 ( .A(transmit), .Y(n9) );
endmodule
```

```
module serialOut_0_DW01_inc_0 ( A, SUM );
  input [29:0] A;
  output [29:0] SUM;

  wire   [29:2] carry;

  HAX1 U1_1_28 ( .A(A[28]), .B(carry[28]), .YC(carry[29]), .YS(SUM[28]) );
  HAX1 U1_1_27 ( .A(A[27]), .B(carry[27]), .YC(carry[28]), .YS(SUM[27]) );
  HAX1 U1_1_26 ( .A(A[26]), .B(carry[26]), .YC(carry[27]), .YS(SUM[26]) );
  HAX1 U1_1_25 ( .A(A[25]), .B(carry[25]), .YC(carry[26]), .YS(SUM[25]) );
  HAX1 U1_1_24 ( .A(A[24]), .B(carry[24]), .YC(carry[25]), .YS(SUM[24]) );
  HAX1 U1_1_23 ( .A(A[23]), .B(carry[23]), .YC(carry[24]), .YS(SUM[23]) );
  HAX1 U1_1_22 ( .A(A[22]), .B(carry[22]), .YC(carry[23]), .YS(SUM[22]) );
  HAX1 U1_1_21 ( .A(A[21]), .B(carry[21]), .YC(carry[22]), .YS(SUM[21]) );
  HAX1 U1_1_20 ( .A(A[20]), .B(carry[20]), .YC(carry[21]), .YS(SUM[20]) );
  HAX1 U1_1_19 ( .A(A[19]), .B(carry[19]), .YC(carry[20]), .YS(SUM[19]) );
  HAX1 U1_1_18 ( .A(A[18]), .B(carry[18]), .YC(carry[19]), .YS(SUM[18]) );
  HAX1 U1_1_17 ( .A(A[17]), .B(carry[17]), .YC(carry[18]), .YS(SUM[17]) );
  HAX1 U1_1_16 ( .A(A[16]), .B(carry[16]), .YC(carry[17]), .YS(SUM[16]) );
  HAX1 U1_1_15 ( .A(A[15]), .B(carry[15]), .YC(carry[16]), .YS(SUM[15]) );
  HAX1 U1_1_14 ( .A(A[14]), .B(carry[14]), .YC(carry[15]), .YS(SUM[14]) );
  HAX1 U1_1_13 ( .A(A[13]), .B(carry[13]), .YC(carry[14]), .YS(SUM[13]) );
  HAX1 U1_1_12 ( .A(A[12]), .B(carry[12]), .YC(carry[13]), .YS(SUM[12]) );
  HAX1 U1_1_11 ( .A(A[11]), .B(carry[11]), .YC(carry[12]), .YS(SUM[11]) );
  HAX1 U1_1_10 ( .A(A[10]), .B(carry[10]), .YC(carry[11]), .YS(SUM[10]) );
  HAX1 U1_1_9 ( .A(A[9]), .B(carry[9]), .YC(carry[10]), .YS(SUM[9]) );
  HAX1 U1_1_8 ( .A(A[8]), .B(carry[8]), .YC(carry[9]), .YS(SUM[8]) );
  HAX1 U1_1_7 ( .A(A[7]), .B(carry[7]), .YC(carry[8]), .YS(SUM[7]) );
  HAX1 U1_1_6 ( .A(A[6]), .B(carry[6]), .YC(carry[7]), .YS(SUM[6]) );
  HAX1 U1_1_5 ( .A(A[5]), .B(carry[5]), .YC(carry[6]), .YS(SUM[5]) );
  HAX1 U1_1_4 ( .A(A[4]), .B(carry[4]), .YC(carry[5]), .YS(SUM[4]) );
  HAX1 U1_1_3 ( .A(A[3]), .B(carry[3]), .YC(carry[4]), .YS(SUM[3]) );
  HAX1 U1_1_2 ( .A(A[2]), .B(carry[2]), .YC(carry[3]), .YS(SUM[2]) );
  HAX1 U1_1_1 ( .A(A[1]), .B(A[0]), .YC(carry[2]), .YS(SUM[1]) );
  XOR2X1 U1 ( .A(carry[29]), .B(A[29]), .Y(SUM[29]) );
  INVX1 U2 ( .A(A[0]), .Y(SUM[0]) );
endmodule
```

```
module serialOut_0 ( clk, rst, initiate, data_in, data_out, trx_busy );
  input [9:0] data_in;
  input clk, rst, initiate;
  output data_out, trx_busy;
  wire   trx_finished_count, transmit, initiate_dly, N44, N45, N46, N47, N48,
         N49, N50, N51, N52, N53, N54, N55, N56, N57, N58, N59, N60, N61, N62,
         N63, N64, N65, N66, N67, N68, N69, N70, N71, N72, N73, N601, N602, n1,
         n2, n3, n4, n5, n6, n7, n8, n9, n10, n11, n12, n13, n14, n15, n16,
         n17, n18, n19, n20, n21, n22, n23, n24, n25, n26, n27, n28, n29, n30,
         n31, n32, n67, n69, n70, n71, n72, n73, n74, n75, n76, n77, n78, n79,
         n80, n81, n82, n83, n84, n85, n86, n87, n88, n89, n90, n91, n92, n93,
         n94, n95, n96, n97, n98, n99, n100, n101, n102, n103, n104, n105,
         n106, n107, n108, n109, n110, n111, n112, n113, n114, n115, n116,
         n117, n118, n119, n120, n121, n122, n123, n124, n125, n126, n127,
         n128, n129, n130, n131, n132, n133, n134, n135, n136, n137, n138,
         n139, n140, n141, n142, n143, n144, n145, n146, n147, n148, n149,
         n150, n151, n152, n153, n154, n155, n156, n157, n158, n159, n160,
         n161, n162, n163, n164, n165, n166, n167, n168, n169, n170, n171,
         n172, n173, n174, n175, n176, n177, n178, n179, n180, n181, n182,
         n183, n184, n185, n186, n187, n188, n189, n190, n191, n192, n193,
         n194, n195, n196, n197, n200, n201, n202, n203, n204, n205, n206,
         n207, n208, n209, n210, n211, n212, n213, n214, n215, n216, n217,
         n218, n219, n220, n221, n222, n223, n224, n225, n226, n227, n228,
         n229, n230, n231, n232, n233, n234, n235, n236, n237, n238, n239,
         n240, n241, n242, n243, n244, n245, n246, n247, n248, n249, n250,
         n251, n252, n253, n254, n255, n256, n257, n258, n259, n260, n261,
         n262, n263, n264, n265, n266, n267, n268, n269, n270, n271, n272,
         n273, n274, n275, n276, n277;
  wire   [29:0] trx_delay_counter;

  DFFSR initiate_dly_reg ( .D(initiate), .CLK(clk), .R(n6), .S(1'b1), .Q(
        initiate_dly) );
  DFFSR trx_busy_reg ( .D(n277), .CLK(clk), .R(n6), .S(1'b1), .Q(trx_busy) );
  DFFSR transmit_reg ( .D(n276), .CLK(clk), .R(n7), .S(1'b1), .Q(transmit) );
  DFFSR trx_delay_counter_reg_29_ ( .D(n246), .CLK(clk), .R(n6), .S(1'b1), .Q(
        trx_delay_counter[29]) );
  DFFSR trx_delay_counter_reg_28_ ( .D(n247), .CLK(clk), .R(n7), .S(1'b1), .Q(
```

```
      trx_delay_counter[28]) );
DFFSR trx_delay_counter_reg_27_ ( .D(n248), .CLK(clk), .R(n6), .S(1'b1), .Q(
      trx_delay_counter[27]) );
DFFSR trx_delay_counter_reg_26_ ( .D(n249), .CLK(clk), .R(n7), .S(1'b1), .Q(
      trx_delay_counter[26]) );
DFFSR trx_delay_counter_reg_25_ ( .D(n250), .CLK(clk), .R(n6), .S(1'b1), .Q(
      trx_delay_counter[25]) );
DFFSR trx_delay_counter_reg_24_ ( .D(n251), .CLK(clk), .R(n7), .S(1'b1), .Q(
      trx_delay_counter[24]) );
DFFSR trx_delay_counter_reg_23_ ( .D(n252), .CLK(clk), .R(n6), .S(1'b1), .Q(
      trx_delay_counter[23]) );
DFFSR trx_delay_counter_reg_22_ ( .D(n253), .CLK(clk), .R(n7), .S(1'b1), .Q(
      trx_delay_counter[22]) );
DFFSR trx_delay_counter_reg_21_ ( .D(n254), .CLK(clk), .R(n7), .S(1'b1), .Q(
      trx_delay_counter[21]) );
DFFSR trx_delay_counter_reg_20_ ( .D(n255), .CLK(clk), .R(n7), .S(1'b1), .Q(
      trx_delay_counter[20]) );
DFFSR trx_delay_counter_reg_19_ ( .D(n256), .CLK(clk), .R(n7), .S(1'b1), .Q(
      trx_delay_counter[19]) );
DFFSR trx_delay_counter_reg_18_ ( .D(n257), .CLK(clk), .R(n7), .S(1'b1), .Q(
      trx_delay_counter[18]) );
DFFSR trx_delay_counter_reg_17_ ( .D(n258), .CLK(clk), .R(n7), .S(1'b1), .Q(
      trx_delay_counter[17]) );
DFFSR trx_delay_counter_reg_16_ ( .D(n259), .CLK(clk), .R(n7), .S(1'b1), .Q(
      trx_delay_counter[16]) );
DFFSR trx_delay_counter_reg_15_ ( .D(n260), .CLK(clk), .R(n7), .S(1'b1), .Q(
      trx_delay_counter[15]) );
DFFSR trx_delay_counter_reg_14_ ( .D(n261), .CLK(clk), .R(n7), .S(1'b1), .Q(
      trx_delay_counter[14]) );
DFFSR trx_delay_counter_reg_13_ ( .D(n262), .CLK(clk), .R(n7), .S(1'b1), .Q(
      trx_delay_counter[13]) );
DFFSR trx_delay_counter_reg_12_ ( .D(n263), .CLK(clk), .R(n7), .S(1'b1), .Q(
      trx_delay_counter[12]) );
DFFSR trx_delay_counter_reg_11_ ( .D(n264), .CLK(clk), .R(n7), .S(1'b1), .Q(
      trx_delay_counter[11]) );
DFFSR trx_delay_counter_reg_10_ ( .D(n265), .CLK(clk), .R(n7), .S(1'b1), .Q(
      trx_delay_counter[10]) );
DFFSR trx_delay_counter_reg_9_ ( .D(n266), .CLK(clk), .R(n6), .S(1'b1), .Q(
```

```
      trx_delay_counter[9]) );
DFFSR trx_delay_counter_reg_8_ ( .D(n267), .CLK(clk), .R(n6), .S(1'b1), .Q(
      trx_delay_counter[8]) );
DFFSR trx_delay_counter_reg_7_ ( .D(n268), .CLK(clk), .R(n6), .S(1'b1), .Q(
      trx_delay_counter[7]) );
DFFSR trx_delay_counter_reg_6_ ( .D(n269), .CLK(clk), .R(n6), .S(1'b1), .Q(
      trx_delay_counter[6]) );
DFFSR trx_delay_counter_reg_5_ ( .D(n270), .CLK(clk), .R(n6), .S(1'b1), .Q(
      trx_delay_counter[5]) );
DFFSR trx_delay_counter_reg_4_ ( .D(n271), .CLK(clk), .R(n6), .S(1'b1), .Q(
      trx_delay_counter[4]) );
DFFSR trx_delay_counter_reg_3_ ( .D(n272), .CLK(clk), .R(n6), .S(1'b1), .Q(
      trx_delay_counter[3]) );
DFFSR trx_delay_counter_reg_2_ ( .D(n273), .CLK(clk), .R(n6), .S(1'b1), .Q(
      trx_delay_counter[2]) );
DFFSR trx_delay_counter_reg_1_ ( .D(n274), .CLK(clk), .R(n6), .S(1'b1), .Q(
      trx_delay_counter[1]) );
DFFSR trx_delay_counter_reg_0_ ( .D(n275), .CLK(clk), .R(n6), .S(1'b1), .Q(
      trx_delay_counter[0]) );
DFFSR trx_finished_count_reg ( .D(N601), .CLK(clk), .R(n6), .S(1'b1), .Q(
      trx_finished_count) );
DFFSR data_out_reg ( .D(N602), .CLK(clk), .R(n6), .S(1'b1), .Q(data_out) );
serialOut_0_DW01_inc_0 r136 ( .A({trx_delay_counter[29:25], n5,
      trx_delay_counter[23:21], n4, trx_delay_counter[19:15], n3, n2,
      trx_delay_counter[12:0]}), .SUM({N73, N72, N71, N70, N69, N68, N67,
      N66, N65, N64, N63, N62, N61, N60, N59, N58, N57, N56, N55, N54, N53,
      N52, N51, N50, N49, N48, N47, N46, N45, N44}) );
AND2X2 U5 ( .A(transmit), .B(n8), .Y(n1) );
INVX2 U6 ( .A(rst), .Y(n6) );
INVX2 U7 ( .A(rst), .Y(n7) );
BUFX2 U8 ( .A(trx_delay_counter[14]), .Y(n3) );
BUFX2 U9 ( .A(trx_delay_counter[13]), .Y(n2) );
BUFX2 U10 ( .A(trx_delay_counter[20]), .Y(n4) );
BUFX2 U11 ( .A(trx_delay_counter[24]), .Y(n5) );
AND2X1 U12 ( .A(N73), .B(n1), .Y(n246) );
AND2X1 U13 ( .A(N72), .B(n1), .Y(n247) );
AND2X1 U14 ( .A(N71), .B(n1), .Y(n248) );
AND2X1 U15 ( .A(N70), .B(n1), .Y(n249) );
```

```
AND2X1 U16 ( .A(N69), .B(n1), .Y(n250) );
AND2X1 U17 ( .A(N68), .B(n1), .Y(n251) );
AND2X1 U18 ( .A(N67), .B(n1), .Y(n252) );
AND2X1 U19 ( .A(N66), .B(n1), .Y(n253) );
AND2X1 U20 ( .A(N65), .B(n1), .Y(n254) );
AND2X1 U53 ( .A(N64), .B(n1), .Y(n255) );
AND2X1 U55 ( .A(N63), .B(n1), .Y(n256) );
AND2X1 U56 ( .A(N62), .B(n1), .Y(n257) );
AND2X1 U57 ( .A(N61), .B(n1), .Y(n258) );
AND2X1 U58 ( .A(N60), .B(n1), .Y(n259) );
AND2X1 U59 ( .A(N59), .B(n1), .Y(n260) );
AND2X1 U60 ( .A(N58), .B(n1), .Y(n261) );
AND2X1 U61 ( .A(N57), .B(n1), .Y(n262) );
AND2X1 U62 ( .A(N56), .B(n1), .Y(n263) );
AND2X1 U63 ( .A(N55), .B(n1), .Y(n264) );
AND2X1 U64 ( .A(N54), .B(n1), .Y(n265) );
AND2X1 U65 ( .A(N53), .B(n1), .Y(n266) );
AND2X1 U66 ( .A(N52), .B(n1), .Y(n267) );
AND2X1 U67 ( .A(N51), .B(n1), .Y(n268) );
AND2X1 U68 ( .A(N50), .B(n1), .Y(n269) );
AND2X1 U69 ( .A(N49), .B(n1), .Y(n270) );
AND2X1 U70 ( .A(N48), .B(n1), .Y(n271) );
AND2X1 U71 ( .A(N47), .B(n1), .Y(n272) );
AND2X1 U72 ( .A(N46), .B(n1), .Y(n273) );
AND2X1 U73 ( .A(N45), .B(n1), .Y(n274) );
AND2X1 U74 ( .A(N44), .B(n1), .Y(n275) );
MUX2X1 U75 ( .B(trx_finished_count), .A(n9), .S(n10), .Y(n276) );
MUX2X1 U76 ( .B(trx_finished_count), .A(n11), .S(n10), .Y(n277) );
AOI21X1 U77 ( .A(n12), .B(initiate), .C(trx_finished_count), .Y(n10) );
INVX1 U78 ( .A(initiate_dly), .Y(n12) );
INVX1 U79 ( .A(trx_busy), .Y(n11) );
AOI21X1 U80 ( .A(n13), .B(n14), .C(n9), .Y(N602) );
NOR2X1 U81 ( .A(n15), .B(n16), .Y(n14) );
OAI22X1 U82 ( .A(n17), .B(n18), .C(n19), .D(n20), .Y(n16) );
MUX2X1 U83 ( .B(n21), .A(data_in[7]), .S(n22), .Y(n20) );
OAI21X1 U84 ( .A(n23), .B(n24), .C(n25), .Y(n22) );
OAI21X1 U85 ( .A(n26), .B(n27), .C(n28), .Y(n24) );
NOR2X1 U86 ( .A(n29), .B(n30), .Y(n26) );
```

```
OAI21X1 U87 ( .A(n31), .B(n32), .C(n67), .Y(n30) );
OAI21X1 U88 ( .A(n69), .B(n70), .C(trx_delay_counter[16]), .Y(n32) );
OAI21X1 U89 ( .A(n71), .B(n72), .C(n73), .Y(n70) );
INVX1 U90 ( .A(n74), .Y(n69) );
OAI21X1 U91 ( .A(trx_delay_counter[10]), .B(n75), .C(n2), .Y(n74) );
NAND2X1 U92 ( .A(n76), .B(n77), .Y(n29) );
AND2X1 U93 ( .A(data_in[6]), .B(n78), .Y(n21) );
NAND2X1 U94 ( .A(data_in[0]), .B(n79), .Y(n18) );
INVX1 U95 ( .A(n80), .Y(n79) );
OAI21X1 U96 ( .A(n81), .B(n82), .C(n83), .Y(n17) );
NOR2X1 U97 ( .A(trx_delay_counter[28]), .B(trx_delay_counter[27]), .Y(n83)
      );
NAND2X1 U98 ( .A(trx_delay_counter[26]), .B(n84), .Y(n82) );
OAI21X1 U99 ( .A(n85), .B(n86), .C(n87), .Y(n84) );
NOR2X1 U100 ( .A(n5), .B(n88), .Y(n87) );
NAND3X1 U101 ( .A(n4), .B(trx_delay_counter[21]), .C(n89), .Y(n86) );
OR2X1 U102 ( .A(n90), .B(n91), .Y(n85) );
OAI21X1 U103 ( .A(n92), .B(n93), .C(trx_delay_counter[23]), .Y(n91) );
OAI21X1 U104 ( .A(n94), .B(n95), .C(n73), .Y(n93) );
OAI21X1 U105 ( .A(n96), .B(trx_delay_counter[11]), .C(n2), .Y(n95) );
AND2X1 U106 ( .A(trx_delay_counter[10]), .B(n97), .Y(n96) );
MUX2X1 U107 ( .B(n98), .A(n99), .S(n100), .Y(n15) );
NOR2X1 U108 ( .A(trx_delay_counter[27]), .B(n25), .Y(n100) );
AOI21X1 U109 ( .A(n81), .B(n101), .C(n102), .Y(n99) );
NAND3X1 U110 ( .A(n103), .B(n104), .C(n4), .Y(n101) );
OAI21X1 U111 ( .A(n105), .B(n106), .C(n107), .Y(n104) );
NOR2X1 U112 ( .A(trx_delay_counter[19]), .B(n89), .Y(n107) );
NAND3X1 U113 ( .A(n3), .B(n2), .C(trx_delay_counter[15]), .Y(n106) );
NAND3X1 U114 ( .A(trx_delay_counter[16]), .B(n108), .C(trx_delay_counter[18]), .Y(n105) );
NAND3X1 U115 ( .A(n71), .B(n109), .C(n110), .Y(n108) );
OAI21X1 U116 ( .A(n111), .B(n112), .C(n113), .Y(n98) );
MUX2X1 U117 ( .B(n114), .A(n115), .S(n116), .Y(n113) );
NOR2X1 U118 ( .A(n25), .B(n117), .Y(n116) );
OAI21X1 U119 ( .A(n27), .B(n118), .C(n102), .Y(n117) );
OAI21X1 U120 ( .A(n119), .B(n4), .C(trx_delay_counter[25]), .Y(n118) );
AOI21X1 U121 ( .A(n120), .B(n121), .C(n90), .Y(n119) );
NAND3X1 U122 ( .A(trx_delay_counter[15]), .B(n3), .C(n122), .Y(n121) );
AOI21X1 U123 ( .A(n123), .B(n110), .C(n124), .Y(n122) );
```

```
NOR2X1 U124 ( .A(trx_delay_counter[10]), .B(trx_delay_counter[9]), .Y(n110)
       );
NOR2X1 U125 ( .A(n2), .B(n125), .Y(n123) );
INVX1 U126 ( .A(n103), .Y(n27) );
INVX1 U127 ( .A(data_in[9]), .Y(n115) );
NAND2X1 U128 ( .A(data_in[8]), .B(n19), .Y(n114) );
INVX1 U129 ( .A(n126), .Y(n19) );
OAI21X1 U130 ( .A(n81), .B(n127), .C(n128), .Y(n126) );
OAI21X1 U131 ( .A(n129), .B(n5), .C(n130), .Y(n127) );
AOI21X1 U132 ( .A(n131), .B(n132), .C(n133), .Y(n129) );
NAND3X1 U133 ( .A(n134), .B(n92), .C(n89), .Y(n132) );
OAI21X1 U134 ( .A(n94), .B(n135), .C(n136), .Y(n134) );
NOR2X1 U135 ( .A(trx_delay_counter[16]), .B(n3), .Y(n136) );
OAI21X1 U136 ( .A(n97), .B(n137), .C(n2), .Y(n135) );
OR2X1 U137 ( .A(trx_delay_counter[10]), .B(trx_delay_counter[11]), .Y(n137)
       );
NOR2X1 U138 ( .A(n138), .B(n139), .Y(n97) );
NOR2X1 U139 ( .A(n4), .B(trx_delay_counter[19]), .Y(n131) );
OAI21X1 U140 ( .A(n77), .B(n140), .C(n141), .Y(n112) );
OAI21X1 U141 ( .A(n142), .B(n143), .C(n144), .Y(n140) );
OAI21X1 U142 ( .A(n31), .B(n145), .C(n67), .Y(n143) );
OAI21X1 U143 ( .A(n146), .B(n147), .C(n3), .Y(n145) );
AND2X1 U144 ( .A(n148), .B(trx_delay_counter[12]), .Y(n146) );
INVX1 U145 ( .A(n28), .Y(n111) );
NOR2X1 U146 ( .A(n149), .B(n150), .Y(n13) );
MUX2X1 U147 ( .B(n151), .A(n152), .S(n153), .Y(n150) );
NAND3X1 U148 ( .A(n154), .B(n155), .C(data_in[5]), .Y(n152) );
INVX1 U149 ( .A(n156), .Y(n154) );
NAND3X1 U150 ( .A(n8), .B(n157), .C(n130), .Y(n151) );
OAI21X1 U151 ( .A(n158), .B(n159), .C(n160), .Y(n157) );
OAI21X1 U152 ( .A(trx_delay_counter[22]), .B(n4), .C(n161), .Y(n159) );
NAND3X1 U153 ( .A(n90), .B(n162), .C(n163), .Y(n161) );
OAI21X1 U154 ( .A(n164), .B(n165), .C(trx_delay_counter[18]), .Y(n163) );
NAND2X1 U155 ( .A(n166), .B(n167), .Y(n165) );
OAI21X1 U156 ( .A(n168), .B(n169), .C(n170), .Y(n164) );
NOR2X1 U157 ( .A(trx_delay_counter[17]), .B(n3), .Y(n170) );
INVX1 U158 ( .A(trx_delay_counter[22]), .Y(n162) );
MUX2X1 U159 ( .B(n171), .A(n172), .S(n173), .Y(n149) );
```

```
AOI21X1 U160 ( .A(n174), .B(n153), .C(n128), .Y(n173) );
NAND2X1 U161 ( .A(trx_delay_counter[27]), .B(n175), .Y(n174) );
OAI21X1 U162 ( .A(n176), .B(n81), .C(n102), .Y(n175) );
AOI21X1 U163 ( .A(n88), .B(n177), .C(n5), .Y(n176) );
OAI21X1 U164 ( .A(n178), .B(n77), .C(n179), .Y(n177) );
AOI21X1 U165 ( .A(trx_delay_counter[17]), .B(n180), .C(n181), .Y(n178) );
INVX1 U166 ( .A(n67), .Y(n181) );
OAI21X1 U167 ( .A(n182), .B(n183), .C(n166), .Y(n180) );
NAND2X1 U168 ( .A(trx_delay_counter[12]), .B(n3), .Y(n183) );
OAI21X1 U169 ( .A(trx_delay_counter[11]), .B(n184), .C(n2), .Y(n182) );
AOI21X1 U170 ( .A(n139), .B(n138), .C(n168), .Y(n184) );
NOR2X1 U171 ( .A(trx_delay_counter[8]), .B(trx_delay_counter[7]), .Y(n139)
      );
MUX2X1 U172 ( .B(n185), .A(data_in[3]), .S(n186), .Y(n172) );
AOI21X1 U173 ( .A(n130), .B(n187), .C(trx_delay_counter[29]), .Y(n186) );
OAI21X1 U174 ( .A(n188), .B(n189), .C(n81), .Y(n187) );
NAND2X1 U175 ( .A(n5), .B(n190), .Y(n189) );
OAI21X1 U176 ( .A(n90), .B(n191), .C(n192), .Y(n190) );
NOR2X1 U177 ( .A(trx_delay_counter[21]), .B(n4), .Y(n192) );
OAI21X1 U178 ( .A(trx_delay_counter[17]), .B(n193), .C(trx_delay_counter[18]), .Y(n191) );
AND2X1 U179 ( .A(trx_delay_counter[16]), .B(n194), .Y(n193) );
OAI21X1 U180 ( .A(n73), .B(n195), .C(n31), .Y(n194) );
OAI21X1 U181 ( .A(n196), .B(n125), .C(n2), .Y(n195) );
AOI21X1 U182 ( .A(n109), .B(n138), .C(n168), .Y(n196) );
MUX2X1 U183 ( .B(n197), .A(n200), .S(n201), .Y(n185) );
OAI21X1 U184 ( .A(n202), .B(n203), .C(trx_delay_counter[29]), .Y(n201) );
OAI21X1 U185 ( .A(n204), .B(n188), .C(n28), .Y(n203) );
NOR2X1 U186 ( .A(trx_delay_counter[26]), .B(trx_delay_counter[25]), .Y(n28)
      );
AOI21X1 U187 ( .A(trx_delay_counter[19]), .B(n205), .C(n206), .Y(n204) );
NAND3X1 U188 ( .A(n76), .B(n120), .C(n207), .Y(n205) );
AOI21X1 U189 ( .A(n3), .B(n208), .C(n92), .Y(n207) );
INVX1 U190 ( .A(n166), .Y(n92) );
NOR2X1 U191 ( .A(trx_delay_counter[16]), .B(trx_delay_counter[15]), .Y(n166)
      );
OAI21X1 U192 ( .A(n168), .B(n209), .C(n167), .Y(n208) );
NAND2X1 U193 ( .A(trx_delay_counter[12]), .B(n148), .Y(n209) );
OAI21X1 U194 ( .A(n210), .B(n109), .C(n138), .Y(n148) );
```

```
INVX1 U195 ( .A(trx_delay_counter[7]), .Y(n210) );
NAND3X1 U196 ( .A(n211), .B(n212), .C(n141), .Y(n202) );
INVX1 U197 ( .A(data_in[2]), .Y(n200) );
NAND2X1 U198 ( .A(data_in[1]), .B(n80), .Y(n197) );
OAI21X1 U199 ( .A(n213), .B(n214), .C(trx_delay_counter[29]), .Y(n80) );
OAI21X1 U200 ( .A(n215), .B(n216), .C(n102), .Y(n214) );
NAND2X1 U201 ( .A(n88), .B(trx_delay_counter[25]), .Y(n216) );
INVX1 U202 ( .A(n188), .Y(n88) );
OAI21X1 U203 ( .A(n206), .B(n217), .C(n5), .Y(n215) );
OAI21X1 U204 ( .A(n218), .B(n124), .C(n67), .Y(n217) );
NOR2X1 U205 ( .A(trx_delay_counter[19]), .B(trx_delay_counter[18]), .Y(n67)
       );
INVX1 U206 ( .A(n219), .Y(n124) );
AND2X1 U207 ( .A(n31), .B(n220), .Y(n218) );
OAI21X1 U208 ( .A(n125), .B(n221), .C(n3), .Y(n220) );
OAI21X1 U209 ( .A(n168), .B(n138), .C(n72), .Y(n221) );
INVX1 U210 ( .A(n71), .Y(n125) );
NAND2X1 U211 ( .A(n77), .B(n179), .Y(n206) );
NAND2X1 U212 ( .A(n211), .B(n212), .Y(n213) );
NAND3X1 U213 ( .A(n155), .B(n156), .C(data_in[4]), .Y(n171) );
OAI21X1 U214 ( .A(n222), .B(n81), .C(n211), .Y(n156) );
INVX1 U215 ( .A(trx_delay_counter[25]), .Y(n81) );
AOI21X1 U216 ( .A(n223), .B(n224), .C(n103), .Y(n222) );
NOR2X1 U217 ( .A(n133), .B(n141), .Y(n103) );
INVX1 U218 ( .A(n144), .Y(n133) );
NOR2X1 U219 ( .A(n179), .B(n188), .Y(n144) );
AOI21X1 U220 ( .A(n90), .B(n225), .C(n141), .Y(n224) );
INVX1 U221 ( .A(n5), .Y(n141) );
NAND3X1 U222 ( .A(n89), .B(trx_delay_counter[15]), .C(n226), .Y(n225) );
INVX1 U223 ( .A(n227), .Y(n226) );
OAI21X1 U224 ( .A(n228), .B(trx_delay_counter[10]), .C(trx_delay_counter[16]), .Y(n227) );
NOR2X1 U225 ( .A(n76), .B(n120), .Y(n89) );
INVX1 U226 ( .A(trx_delay_counter[18]), .Y(n120) );
INVX1 U227 ( .A(trx_delay_counter[19]), .Y(n90) );
NOR2X1 U228 ( .A(n188), .B(n77), .Y(n223) );
INVX1 U229 ( .A(n4), .Y(n77) );
NAND2X1 U230 ( .A(trx_delay_counter[22]), .B(trx_delay_counter[23]), .Y(n188) );
INVX1 U231 ( .A(n78), .Y(n155) );
```

```
OAI21X1 U232 ( .A(n229), .B(n23), .C(n25), .Y(n78) );
INVX1 U233 ( .A(n128), .Y(n25) );
NOR2X1 U234 ( .A(trx_delay_counter[28]), .B(trx_delay_counter[29]), .Y(n128)
      );
NAND2X1 U235 ( .A(n211), .B(n153), .Y(n23) );
INVX1 U236 ( .A(trx_delay_counter[29]), .Y(n153) );
AOI21X1 U237 ( .A(n160), .B(n230), .C(n102), .Y(n229) );
NAND3X1 U238 ( .A(trx_delay_counter[23]), .B(n231), .C(trx_delay_counter[22]), .Y(n230) );
OAI21X1 U239 ( .A(n232), .B(n233), .C(n179), .Y(n231) );
INVX1 U240 ( .A(trx_delay_counter[21]), .Y(n179) );
NAND2X1 U241 ( .A(n4), .B(trx_delay_counter[19]), .Y(n233) );
OAI21X1 U242 ( .A(n234), .B(n142), .C(trx_delay_counter[18]), .Y(n232) );
NAND2X1 U243 ( .A(n235), .B(n76), .Y(n142) );
AOI21X1 U244 ( .A(n236), .B(n237), .C(n31), .Y(n234) );
INVX1 U245 ( .A(trx_delay_counter[15]), .Y(n31) );
INVX1 U246 ( .A(n147), .Y(n237) );
OAI21X1 U247 ( .A(n94), .B(n168), .C(n167), .Y(n147) );
AOI21X1 U248 ( .A(trx_delay_counter[11]), .B(trx_delay_counter[12]), .C(n2),
     .Y(n167) );
INVX1 U249 ( .A(trx_delay_counter[10]), .Y(n168) );
INVX1 U250 ( .A(trx_delay_counter[12]), .Y(n94) );
AND2X1 U251 ( .A(n73), .B(n169), .Y(n236) );
NAND3X1 U252 ( .A(trx_delay_counter[7]), .B(trx_delay_counter[12]), .C(n75),
     .Y(n169) );
NOR2X1 U253 ( .A(n138), .B(n109), .Y(n75) );
INVX1 U254 ( .A(trx_delay_counter[8]), .Y(n109) );
INVX1 U255 ( .A(trx_delay_counter[9]), .Y(n138) );
NOR2X1 U256 ( .A(trx_delay_counter[25]), .B(n5), .Y(n160) );
NOR2X1 U257 ( .A(n9), .B(n8), .Y(N601) );
NAND2X1 U258 ( .A(trx_delay_counter[29]), .B(n238), .Y(n8) );
OAI21X1 U259 ( .A(n239), .B(n240), .C(n212), .Y(n238) );
INVX1 U260 ( .A(trx_delay_counter[28]), .Y(n212) );
NAND2X1 U261 ( .A(trx_delay_counter[25]), .B(n5), .Y(n240) );
NAND3X1 U262 ( .A(n241), .B(n242), .C(n130), .Y(n239) );
NOR2X1 U263 ( .A(n211), .B(n102), .Y(n130) );
INVX1 U264 ( .A(trx_delay_counter[26]), .Y(n102) );
INVX1 U265 ( .A(trx_delay_counter[27]), .Y(n211) );
OAI21X1 U266 ( .A(n243), .B(n244), .C(n245), .Y(n242) );
```

```
  NOR2X1 U267 ( .A(trx_delay_counter[22]), .B(n4), .Y(n245) );
  NAND2X1 U268 ( .A(trx_delay_counter[19]), .B(trx_delay_counter[18]), .Y(n244) );
  OAI21X1 U269 ( .A(trx_delay_counter[15]), .B(n228), .C(n219), .Y(n243) );
  NOR2X1 U270 ( .A(n76), .B(n235), .Y(n219) );
  INVX1 U271 ( .A(trx_delay_counter[16]), .Y(n235) );
  INVX1 U272 ( .A(trx_delay_counter[17]), .Y(n76) );
  NAND3X1 U273 ( .A(n72), .B(n73), .C(n71), .Y(n228) );
  NOR2X1 U274 ( .A(trx_delay_counter[12]), .B(trx_delay_counter[11]), .Y(n71)
        );
  INVX1 U275 ( .A(n3), .Y(n73) );
  INVX1 U276 ( .A(n2), .Y(n72) );
  INVX1 U277 ( .A(n158), .Y(n241) );
  OAI21X1 U278 ( .A(trx_delay_counter[22]), .B(trx_delay_counter[21]), .C(
       trx_delay_counter[23]), .Y(n158) );
  INVX1 U279 ( .A(transmit), .Y(n9) );
endmodule


module daq_DW01_inc_0 ( A, SUM );
  input [9:0] A;
  output [9:0] SUM;

  wire   [9:2] carry;

  HAX1 U1_1_8 ( .A(A[8]), .B(carry[8]), .YC(carry[9]), .YS(SUM[8]) );
  HAX1 U1_1_7 ( .A(A[7]), .B(carry[7]), .YC(carry[8]), .YS(SUM[7]) );
  HAX1 U1_1_6 ( .A(A[6]), .B(carry[6]), .YC(carry[7]), .YS(SUM[6]) );
  HAX1 U1_1_5 ( .A(A[5]), .B(carry[5]), .YC(carry[6]), .YS(SUM[5]) );
  HAX1 U1_1_4 ( .A(A[4]), .B(carry[4]), .YC(carry[5]), .YS(SUM[4]) );
  HAX1 U1_1_3 ( .A(A[3]), .B(carry[3]), .YC(carry[4]), .YS(SUM[3]) );
  HAX1 U1_1_2 ( .A(A[2]), .B(carry[2]), .YC(carry[3]), .YS(SUM[2]) );
  HAX1 U1_1_1 ( .A(A[1]), .B(A[0]), .YC(carry[2]), .YS(SUM[1]) );
  XOR2X1 U1 ( .A(carry[9]), .B(A[9]), .Y(SUM[9]) );
  INVX1 U2 ( .A(A[0]), .Y(SUM[0]) );
endmodule


module daq_DW01_inc_1 ( A, SUM );
```

```
  input [9:0] A;
  output [9:0] SUM;

  wire   [9:2] carry;

  HAX1 U1_1_8 ( .A(A[8]), .B(carry[8]), .YC(carry[9]), .YS(SUM[8]) );
  HAX1 U1_1_7 ( .A(A[7]), .B(carry[7]), .YC(carry[8]), .YS(SUM[7]) );
  HAX1 U1_1_6 ( .A(A[6]), .B(carry[6]), .YC(carry[7]), .YS(SUM[6]) );
  HAX1 U1_1_5 ( .A(A[5]), .B(carry[5]), .YC(carry[6]), .YS(SUM[5]) );
  HAX1 U1_1_4 ( .A(A[4]), .B(carry[4]), .YC(carry[5]), .YS(SUM[4]) );
  HAX1 U1_1_3 ( .A(A[3]), .B(carry[3]), .YC(carry[4]), .YS(SUM[3]) );
  HAX1 U1_1_2 ( .A(A[2]), .B(carry[2]), .YC(carry[3]), .YS(SUM[2]) );
  HAX1 U1_1_1 ( .A(A[1]), .B(A[0]), .YC(carry[2]), .YS(SUM[1]) );
  XOR2X1 U1 ( .A(carry[9]), .B(A[9]), .Y(SUM[9]) );
  INVX1 U2 ( .A(A[0]), .Y(SUM[0]) );
endmodule


module daq_DW01_inc_2 ( A, SUM );
  input [9:0] A;
  output [9:0] SUM;

  wire   [9:2] carry;

  HAX1 U1_1_8 ( .A(A[8]), .B(carry[8]), .YC(carry[9]), .YS(SUM[8]) );
  HAX1 U1_1_7 ( .A(A[7]), .B(carry[7]), .YC(carry[8]), .YS(SUM[7]) );
  HAX1 U1_1_6 ( .A(A[6]), .B(carry[6]), .YC(carry[7]), .YS(SUM[6]) );
  HAX1 U1_1_5 ( .A(A[5]), .B(carry[5]), .YC(carry[6]), .YS(SUM[5]) );
  HAX1 U1_1_4 ( .A(A[4]), .B(carry[4]), .YC(carry[5]), .YS(SUM[4]) );
  HAX1 U1_1_3 ( .A(A[3]), .B(carry[3]), .YC(carry[4]), .YS(SUM[3]) );
  HAX1 U1_1_2 ( .A(A[2]), .B(carry[2]), .YC(carry[3]), .YS(SUM[2]) );
  HAX1 U1_1_1 ( .A(A[1]), .B(A[0]), .YC(carry[2]), .YS(SUM[1]) );
  XOR2X1 U1 ( .A(carry[9]), .B(A[9]), .Y(SUM[9]) );
  INVX1 U2 ( .A(A[0]), .Y(SUM[0]) );
endmodule


module daq_DW01_inc_3 ( A, SUM );
```

```
  input [29:0] A;
  output [29:0] SUM;

  wire   [29:2] carry;

  HAX1 U1_1_28 ( .A(A[28]), .B(carry[28]), .YC(carry[29]), .YS(SUM[28]) );
  HAX1 U1_1_27 ( .A(A[27]), .B(carry[27]), .YC(carry[28]), .YS(SUM[27]) );
  HAX1 U1_1_26 ( .A(A[26]), .B(carry[26]), .YC(carry[27]), .YS(SUM[26]) );
  HAX1 U1_1_25 ( .A(A[25]), .B(carry[25]), .YC(carry[26]), .YS(SUM[25]) );
  HAX1 U1_1_24 ( .A(A[24]), .B(carry[24]), .YC(carry[25]), .YS(SUM[24]) );
  HAX1 U1_1_23 ( .A(A[23]), .B(carry[23]), .YC(carry[24]), .YS(SUM[23]) );
  HAX1 U1_1_22 ( .A(A[22]), .B(carry[22]), .YC(carry[23]), .YS(SUM[22]) );
  HAX1 U1_1_21 ( .A(A[21]), .B(carry[21]), .YC(carry[22]), .YS(SUM[21]) );
  HAX1 U1_1_20 ( .A(A[20]), .B(carry[20]), .YC(carry[21]), .YS(SUM[20]) );
  HAX1 U1_1_19 ( .A(A[19]), .B(carry[19]), .YC(carry[20]), .YS(SUM[19]) );
  HAX1 U1_1_18 ( .A(A[18]), .B(carry[18]), .YC(carry[19]), .YS(SUM[18]) );
  HAX1 U1_1_17 ( .A(A[17]), .B(carry[17]), .YC(carry[18]), .YS(SUM[17]) );
  HAX1 U1_1_16 ( .A(A[16]), .B(carry[16]), .YC(carry[17]), .YS(SUM[16]) );
  HAX1 U1_1_15 ( .A(A[15]), .B(carry[15]), .YC(carry[16]), .YS(SUM[15]) );
  HAX1 U1_1_14 ( .A(A[14]), .B(carry[14]), .YC(carry[15]), .YS(SUM[14]) );
  HAX1 U1_1_13 ( .A(A[13]), .B(carry[13]), .YC(carry[14]), .YS(SUM[13]) );
  HAX1 U1_1_12 ( .A(A[12]), .B(carry[12]), .YC(carry[13]), .YS(SUM[12]) );
  HAX1 U1_1_11 ( .A(A[11]), .B(carry[11]), .YC(carry[12]), .YS(SUM[11]) );
  HAX1 U1_1_10 ( .A(A[10]), .B(carry[10]), .YC(carry[11]), .YS(SUM[10]) );
  HAX1 U1_1_9 ( .A(A[9]), .B(carry[9]), .YC(carry[10]), .YS(SUM[9]) );
  HAX1 U1_1_8 ( .A(A[8]), .B(carry[8]), .YC(carry[9]), .YS(SUM[8]) );
  HAX1 U1_1_7 ( .A(A[7]), .B(carry[7]), .YC(carry[8]), .YS(SUM[7]) );
  HAX1 U1_1_6 ( .A(A[6]), .B(carry[6]), .YC(carry[7]), .YS(SUM[6]) );
  HAX1 U1_1_5 ( .A(A[5]), .B(carry[5]), .YC(carry[6]), .YS(SUM[5]) );
  HAX1 U1_1_4 ( .A(A[4]), .B(carry[4]), .YC(carry[5]), .YS(SUM[4]) );
  HAX1 U1_1_3 ( .A(A[3]), .B(carry[3]), .YC(carry[4]), .YS(SUM[3]) );
  HAX1 U1_1_2 ( .A(A[2]), .B(carry[2]), .YC(carry[3]), .YS(SUM[2]) );
  HAX1 U1_1_1 ( .A(A[1]), .B(A[0]), .YC(carry[2]), .YS(SUM[1]) );
  XOR2X1 U1 ( .A(carry[29]), .B(A[29]), .Y(SUM[29]) );
  INVX1 U2 ( .A(A[0]), .Y(SUM[0]) );
endmodule
```

```
module daq ( paddle_in_n, startCount_n, stopCount_n, rst, clk,
        sendSerialData_n, serialCount_trx_busy, serialDeadCount_trx_busy,
        runStateIndicator, pausedIndicator, display_totalCounts_out,
        display_counts_out, display_deadCounts_out, serial_count_out,
        serial_deadCount_out, counts, deadCounts, totalCounts, busy );
  input [2:0] paddle_in_n;
  output [20:0] display_totalCounts_out;
  output [20:0] display_counts_out;
  output [13:0] display_deadCounts_out;
  output [9:0] counts;
  output [9:0] deadCounts;
  output [9:0] totalCounts;
  input startCount_n, stopCount_n, rst, clk, sendSerialData_n;
  output serialCount_trx_busy, serialDeadCount_trx_busy, runStateIndicator,
         pausedIndicator, serial_count_out, serial_deadCount_out, busy;
  wire   trigger_dly, start_latch_counter, trigger_latched, finishedCount, N36,
         N37, N38, N45, N47, N48, N49, N50, N51, N52, N53, N54, N55, N56, N57,
         N58, N59, N60, N61, N62, N63, N64, N65, N66, N67, N68, N69, N70, N71,
         N72, N73, N74, N75, N76, N107, N108, N109, N110, N111, N112, N113,
         N114, N115, N116, N117, N118, N119, N120, N121, N122, N123, N124,
         N125, N126, N127, N128, N129, N130, N131, N132, N133, N134, N135,
         N136, N137, trigger_latched_dly, startCount_n_dly, stopCount_n_dly,
         N149, N150, N151, N152, N153, N154, N155, N156, N157, N158, N213,
         N214, N215, N216, N217, N218, N219, N220, N221, N222, N263, N264,
         N265, N266, N267, N268, N269, N270, N271, N272, sendSerialData_dly,
         N336, N337, N338, N339, N340, N341, N342, N343, N344, N345, N346,
         N347, N348, N349, N350, N351, N352, N353, N354, N355, n226, n227,
         n228, n229, n230, n231, n232, n233, n234, n235, n236, n237, n238,
         n239, n240, n241, n242, n243, n244, n245, n246, n247, n248, n249,
         n250, n251, n252, n253, n254, n255, n256, n257, n258, n259, n260,
         n261, n262, n263, n264, n265, n266, n267, n268, n269, n270, n271,
         n272, n273, n274, n275, n276, n277, n278, n279, n280, n281, n282,
         n283, n285, n286, n287, n288, n289, n290, n291, n292, n293, n294,
         n295, n296, n297, n298, n299, n300, n301, n302, n303, n304, n305,
         n306, n307, n308, n309, n310, n311, n312, n313, n314, n315, n316,
         n317, n318, n319, n320, n321, n322, n323, n324, n325, n326, n327,
         n328, n329, n330, n331, n332, n333, n334, n335, n336, n337, n338,
         n339, n340, n341, n342, n343, n345, n346, n347, n348, n349, n350,
```

```
        n351, n352, n353, n354, n355, n356, n357, n358, n359, n360, n361,
        n362, n363, n364, n365, n366, n368, n369, n370, n371, n372, n373,
        n374, n375, n376, n377, n378, n379, n380, n381, n382, n383, n384,
        n385, n386, n387, n388, n389, n390, n391, n392, n393, n394, n395,
        n396, n397, n398, n399, n400, n401, n402, n403, n404, n405, n406,
        n407, n408, n409, n410, n411, n412, n413, n414, n415, n416, n417,
        n418, n419, n1, n2, n3, n4, n127, n129, n130, n131, n132, n133, n134,
        n135, n136, n137, n138, n139, n140, n141, n142, n143, n144, n145,
        n146, n147, n148, n149, n150, n151, n152, n153, n154, n155, n156,
        n157, n158, n159, n160, n161, n162, n163, n164, n165, n166, n167,
        n168, n169, n170, n171, n172, n173, n174, n175, n176, n177, n178,
        n179, n180, n181, n182, n183, n184, n185, n186, n187, n188, n189,
        n190, n191, n192, n193, n194, n195, n196, n197, n198, n199, n200,
        n201, n202, n203, n204, n205, n206, n207, n208, n209, n210, n211,
        n212, n213, n214, n215, n216, n217, n218, n219, n220, n221, n222,
        n223, n224, n225, n284, n344, n367, n420, n421, n422, n423, n424,
        n425, n426, n427, n428, n429, n430, n431, n432, n433, n434, n435,
        n436, n437, n438, n439, n440, n441, n442, n443, n444, n445, n446,
        n447, n448, n449, n450, n451, n452;
  wire   [29:0] latch_counter;
  wire   [1:0] systemState;
  wire   [11:0] totalCounts_bcd;
  wire   [11:0] counts_bcd;
  wire   [7:0] deadCounts_bcd;
  wire   [9:0] count_captured;
  wire   [9:0] deadCount_captured;

  DFFSR trigger_dly_reg ( .D(n419), .CLK(clk), .R(n134), .S(1'b1), .Q(
        trigger_dly) );
  DFFSR startCount_n_dly_reg ( .D(startCount_n), .CLK(clk), .R(1'b1), .S(n139),
        .Q(startCount_n_dly) );
  DFFSR stopCount_n_dly_reg ( .D(stopCount_n), .CLK(clk), .R(1'b1), .S(n139),
        .Q(stopCount_n_dly) );
  DFFSR systemState_reg_1_ ( .D(n190), .CLK(clk), .R(n129), .S(1'b1), .Q(
        systemState[1]) );
  DFFSR systemState_reg_0_ ( .D(n418), .CLK(clk), .R(n129), .S(1'b1), .Q(
        systemState[0]) );
  DFFSR busy_reg ( .D(N38), .CLK(clk), .R(n129), .S(1'b1), .Q(busy) );
```

```
DFFSR trigger_latched_reg ( .D(N37), .CLK(clk), .R(n129), .S(1'b1), .Q(
      trigger_latched) );
DFFSR trigger_latched_dly_reg ( .D(trigger_latched), .CLK(clk), .R(n129),
      .S(1'b1), .Q(trigger_latched_dly) );
DFFSR start_latch_counter_reg ( .D(N36), .CLK(clk), .R(n129), .S(1'b1), .Q(
      start_latch_counter) );
DFFSR latch_counter_reg_0_ ( .D(N108), .CLK(clk), .R(n129), .S(1'b1), .Q(
      latch_counter[0]) );
DFFSR latch_counter_reg_1_ ( .D(N109), .CLK(clk), .R(n129), .S(1'b1), .Q(
      latch_counter[1]) );
DFFSR latch_counter_reg_2_ ( .D(N110), .CLK(clk), .R(n129), .S(1'b1), .Q(
      latch_counter[2]) );
DFFSR latch_counter_reg_3_ ( .D(N111), .CLK(clk), .R(n129), .S(1'b1), .Q(
      latch_counter[3]) );
DFFSR latch_counter_reg_4_ ( .D(N112), .CLK(clk), .R(n129), .S(1'b1), .Q(
      latch_counter[4]) );
DFFSR latch_counter_reg_5_ ( .D(N113), .CLK(clk), .R(n130), .S(1'b1), .Q(
      latch_counter[5]) );
DFFSR latch_counter_reg_6_ ( .D(N114), .CLK(clk), .R(n130), .S(1'b1), .Q(
      latch_counter[6]) );
DFFSR latch_counter_reg_7_ ( .D(N115), .CLK(clk), .R(n130), .S(1'b1), .Q(
      latch_counter[7]) );
DFFSR latch_counter_reg_8_ ( .D(N116), .CLK(clk), .R(n130), .S(1'b1), .Q(
      latch_counter[8]) );
DFFSR latch_counter_reg_9_ ( .D(N117), .CLK(clk), .R(n130), .S(1'b1), .Q(
      latch_counter[9]) );
DFFSR latch_counter_reg_10_ ( .D(N118), .CLK(clk), .R(n130), .S(1'b1), .Q(
      latch_counter[10]) );
DFFSR latch_counter_reg_11_ ( .D(N119), .CLK(clk), .R(n130), .S(1'b1), .Q(
      latch_counter[11]) );
DFFSR latch_counter_reg_12_ ( .D(N120), .CLK(clk), .R(n130), .S(1'b1), .Q(
      latch_counter[12]) );
DFFSR latch_counter_reg_13_ ( .D(N121), .CLK(clk), .R(n130), .S(1'b1), .Q(
      latch_counter[13]) );
DFFSR latch_counter_reg_14_ ( .D(N122), .CLK(clk), .R(n130), .S(1'b1), .Q(
      latch_counter[14]) );
DFFSR latch_counter_reg_15_ ( .D(N123), .CLK(clk), .R(n130), .S(1'b1), .Q(
      latch_counter[15]) );
```

```
DFFSR latch_counter_reg_16_ ( .D(N124), .CLK(clk), .R(n130), .S(1'b1), .Q(
      latch_counter[16]) );
DFFSR latch_counter_reg_17_ ( .D(N125), .CLK(clk), .R(n131), .S(1'b1), .Q(
      latch_counter[17]) );
DFFSR latch_counter_reg_18_ ( .D(N126), .CLK(clk), .R(n131), .S(1'b1), .Q(
      latch_counter[18]) );
DFFSR latch_counter_reg_19_ ( .D(N127), .CLK(clk), .R(n131), .S(1'b1), .Q(
      latch_counter[19]) );
DFFSR latch_counter_reg_20_ ( .D(N128), .CLK(clk), .R(n131), .S(1'b1), .Q(
      latch_counter[20]) );
DFFSR latch_counter_reg_21_ ( .D(N129), .CLK(clk), .R(n131), .S(1'b1), .Q(
      latch_counter[21]) );
DFFSR latch_counter_reg_22_ ( .D(N130), .CLK(clk), .R(n131), .S(1'b1), .Q(
      latch_counter[22]) );
DFFSR latch_counter_reg_23_ ( .D(N131), .CLK(clk), .R(n131), .S(1'b1), .Q(
      latch_counter[23]) );
DFFSR latch_counter_reg_24_ ( .D(N132), .CLK(clk), .R(n131), .S(1'b1), .Q(
      latch_counter[24]) );
DFFSR latch_counter_reg_25_ ( .D(N133), .CLK(clk), .R(n131), .S(1'b1), .Q(
      latch_counter[25]) );
DFFSR latch_counter_reg_26_ ( .D(N134), .CLK(clk), .R(n131), .S(1'b1), .Q(
      latch_counter[26]) );
DFFSR latch_counter_reg_27_ ( .D(N135), .CLK(clk), .R(n131), .S(1'b1), .Q(
      latch_counter[27]) );
DFFSR latch_counter_reg_28_ ( .D(N136), .CLK(clk), .R(n131), .S(1'b1), .Q(
      latch_counter[28]) );
DFFSR latch_counter_reg_29_ ( .D(N137), .CLK(clk), .R(n132), .S(1'b1), .Q(
      latch_counter[29]) );
DFFSR finishedCount_reg ( .D(N107), .CLK(clk), .R(n132), .S(1'b1), .Q(
      finishedCount) );
DFFSR counts_reg_0_ ( .D(n417), .CLK(clk), .R(n132), .S(1'b1), .Q(counts[0])
       );
DFFSR counts_reg_1_ ( .D(n416), .CLK(clk), .R(n132), .S(1'b1), .Q(counts[1])
       );
DFFSR counts_reg_2_ ( .D(n415), .CLK(clk), .R(n132), .S(1'b1), .Q(counts[2])
       );
DFFSR counts_reg_3_ ( .D(n414), .CLK(clk), .R(n132), .S(1'b1), .Q(counts[3])
       );
```

```
DFFSR counts_reg_4_ ( .D(n413), .CLK(clk), .R(n132), .S(1'b1), .Q(counts[4])
       );
DFFSR counts_reg_5_ ( .D(n412), .CLK(clk), .R(n132), .S(1'b1), .Q(counts[5])
       );
DFFSR counts_reg_6_ ( .D(n411), .CLK(clk), .R(n132), .S(1'b1), .Q(counts[6])
       );
DFFSR counts_reg_7_ ( .D(n410), .CLK(clk), .R(n132), .S(1'b1), .Q(counts[7])
       );
DFFSR counts_reg_8_ ( .D(n409), .CLK(clk), .R(n132), .S(1'b1), .Q(counts[8])
       );
DFFSR counts_reg_9_ ( .D(n408), .CLK(clk), .R(n132), .S(1'b1), .Q(counts[9])
       );
DFFSR deadCounts_bcd_reg_0_ ( .D(n184), .CLK(clk), .R(n133), .S(1'b1), .Q(
      deadCounts_bcd[0]) );
DFFSR deadCounts_bcd_reg_1_ ( .D(n407), .CLK(clk), .R(n133), .S(1'b1), .Q(
      deadCounts_bcd[1]) );
DFFSR deadCounts_bcd_reg_3_ ( .D(n405), .CLK(clk), .R(n133), .S(1'b1), .Q(
      deadCounts_bcd[3]) );
DFFSR deadCounts_bcd_reg_4_ ( .D(n404), .CLK(clk), .R(n133), .S(1'b1), .Q(
      deadCounts_bcd[4]) );
DFFSR deadCounts_bcd_reg_5_ ( .D(n403), .CLK(clk), .R(n133), .S(1'b1), .Q(
      deadCounts_bcd[5]) );
DFFSR deadCounts_bcd_reg_6_ ( .D(n402), .CLK(clk), .R(n133), .S(1'b1), .Q(
      deadCounts_bcd[6]) );
DFFSR deadCounts_bcd_reg_7_ ( .D(n401), .CLK(clk), .R(n133), .S(1'b1), .Q(
      deadCounts_bcd[7]) );
DFFSR deadCounts_bcd_reg_2_ ( .D(n406), .CLK(clk), .R(n133), .S(1'b1), .Q(
      deadCounts_bcd[2]) );
DFFSR deadCounts_reg_0_ ( .D(n400), .CLK(clk), .R(n133), .S(1'b1), .Q(
      deadCounts[0]) );
DFFSR deadCounts_reg_1_ ( .D(n399), .CLK(clk), .R(n133), .S(1'b1), .Q(
      deadCounts[1]) );
DFFSR deadCounts_reg_2_ ( .D(n398), .CLK(clk), .R(n133), .S(1'b1), .Q(
      deadCounts[2]) );
DFFSR deadCounts_reg_3_ ( .D(n397), .CLK(clk), .R(n133), .S(1'b1), .Q(
      deadCounts[3]) );
DFFSR deadCounts_reg_4_ ( .D(n396), .CLK(clk), .R(n134), .S(1'b1), .Q(
      deadCounts[4]) );
```

```
DFFSR deadCounts_reg_5_ ( .D(n395), .CLK(clk), .R(n134), .S(1'b1), .Q(
      deadCounts[5]) );
DFFSR deadCounts_reg_6_ ( .D(n394), .CLK(clk), .R(n134), .S(1'b1), .Q(
      deadCounts[6]) );
DFFSR deadCounts_reg_7_ ( .D(n393), .CLK(clk), .R(n134), .S(1'b1), .Q(
      deadCounts[7]) );
DFFSR deadCounts_reg_8_ ( .D(n392), .CLK(clk), .R(n134), .S(1'b1), .Q(
      deadCounts[8]) );
DFFSR deadCounts_reg_9_ ( .D(n391), .CLK(clk), .R(n134), .S(1'b1), .Q(
      deadCounts[9]) );
DFFSR counts_bcd_reg_0_ ( .D(n390), .CLK(clk), .R(n134), .S(1'b1), .Q(
      counts_bcd[0]) );
DFFSR counts_bcd_reg_1_ ( .D(n389), .CLK(clk), .R(n134), .S(1'b1), .Q(
      counts_bcd[1]) );
DFFSR counts_bcd_reg_3_ ( .D(n387), .CLK(clk), .R(n134), .S(1'b1), .Q(
      counts_bcd[3]) );
DFFSR counts_bcd_reg_4_ ( .D(n385), .CLK(clk), .R(n134), .S(1'b1), .Q(
      counts_bcd[4]) );
DFFSR counts_bcd_reg_5_ ( .D(n384), .CLK(clk), .R(n134), .S(1'b1), .Q(
      counts_bcd[5]) );
DFFSR counts_bcd_reg_7_ ( .D(n382), .CLK(clk), .R(n135), .S(1'b1), .Q(
      counts_bcd[7]) );
DFFSR counts_bcd_reg_6_ ( .D(n383), .CLK(clk), .R(n135), .S(1'b1), .Q(
      counts_bcd[6]) );
DFFSR counts_bcd_reg_2_ ( .D(n388), .CLK(clk), .R(n135), .S(1'b1), .Q(
      counts_bcd[2]) );
DFFSR counts_bcd_reg_8_ ( .D(n381), .CLK(clk), .R(n135), .S(1'b1), .Q(
      counts_bcd[8]) );
DFFSR counts_bcd_reg_9_ ( .D(n386), .CLK(clk), .R(n135), .S(1'b1), .Q(
      counts_bcd[9]) );
DFFSR counts_bcd_reg_10_ ( .D(n380), .CLK(clk), .R(n135), .S(1'b1), .Q(
      counts_bcd[10]) );
DFFSR counts_bcd_reg_11_ ( .D(n379), .CLK(clk), .R(n135), .S(1'b1), .Q(
      counts_bcd[11]) );
DFFSR totalCounts_bcd_reg_0_ ( .D(n178), .CLK(clk), .R(n135), .S(1'b1), .Q(
      totalCounts_bcd[0]) );
DFFSR totalCounts_bcd_reg_1_ ( .D(n378), .CLK(clk), .R(n135), .S(1'b1), .Q(
      totalCounts_bcd[1]) );
```

```
DFFSR totalCounts_bcd_reg_3_ ( .D(n376), .CLK(clk), .R(n135), .S(1'b1), .Q(
      totalCounts_bcd[3]) );
DFFSR totalCounts_bcd_reg_4_ ( .D(n374), .CLK(clk), .R(n135), .S(1'b1), .Q(
      totalCounts_bcd[4]) );
DFFSR totalCounts_bcd_reg_6_ ( .D(n372), .CLK(clk), .R(n135), .S(1'b1), .Q(
      totalCounts_bcd[6]) );
DFFSR totalCounts_bcd_reg_7_ ( .D(n375), .CLK(clk), .R(n136), .S(1'b1), .Q(
      totalCounts_bcd[7]) );
DFFSR totalCounts_bcd_reg_5_ ( .D(n373), .CLK(clk), .R(n136), .S(1'b1), .Q(
      totalCounts_bcd[5]) );
DFFSR totalCounts_bcd_reg_2_ ( .D(n377), .CLK(clk), .R(n136), .S(1'b1), .Q(
      totalCounts_bcd[2]) );
DFFSR totalCounts_bcd_reg_8_ ( .D(n371), .CLK(clk), .R(n136), .S(1'b1), .Q(
      totalCounts_bcd[8]) );
DFFSR totalCounts_bcd_reg_9_ ( .D(n370), .CLK(clk), .R(n136), .S(1'b1), .Q(
      totalCounts_bcd[9]) );
DFFSR totalCounts_bcd_reg_10_ ( .D(n369), .CLK(clk), .R(n136), .S(1'b1), .Q(
      totalCounts_bcd[10]) );
DFFSR totalCounts_bcd_reg_11_ ( .D(n368), .CLK(clk), .R(n136), .S(1'b1), .Q(
      totalCounts_bcd[11]) );
DFFSR totalCounts_reg_0_ ( .D(n168), .CLK(clk), .R(n136), .S(1'b1), .Q(
      totalCounts[0]) );
DFFSR totalCounts_reg_1_ ( .D(n169), .CLK(clk), .R(n136), .S(1'b1), .Q(
      totalCounts[1]) );
DFFSR totalCounts_reg_2_ ( .D(n170), .CLK(clk), .R(n136), .S(1'b1), .Q(
      totalCounts[2]) );
DFFSR totalCounts_reg_3_ ( .D(n171), .CLK(clk), .R(n136), .S(1'b1), .Q(
      totalCounts[3]) );
DFFSR totalCounts_reg_4_ ( .D(n172), .CLK(clk), .R(n136), .S(1'b1), .Q(
      totalCounts[4]) );
DFFSR totalCounts_reg_5_ ( .D(n173), .CLK(clk), .R(n137), .S(1'b1), .Q(
      totalCounts[5]) );
DFFSR totalCounts_reg_6_ ( .D(n174), .CLK(clk), .R(n137), .S(1'b1), .Q(
      totalCounts[6]) );
DFFSR totalCounts_reg_7_ ( .D(n175), .CLK(clk), .R(n137), .S(1'b1), .Q(
      totalCounts[7]) );
DFFSR totalCounts_reg_8_ ( .D(n176), .CLK(clk), .R(n137), .S(1'b1), .Q(
      totalCounts[8]) );
```

```
DFFSR totalCounts_reg_9_ ( .D(n177), .CLK(clk), .R(n137), .S(1'b1), .Q(
      totalCounts[9]) );
DFFSR sendSerialData_dly_reg ( .D(n226), .CLK(clk), .R(n137), .S(1'b1), .Q(
      sendSerialData_dly) );
DFFSR deadCount_captured_reg_9_ ( .D(N355), .CLK(clk), .R(n137), .S(1'b1),
      .Q(deadCount_captured[9]) );
DFFSR deadCount_captured_reg_8_ ( .D(N354), .CLK(clk), .R(n137), .S(1'b1),
      .Q(deadCount_captured[8]) );
DFFSR deadCount_captured_reg_7_ ( .D(N353), .CLK(clk), .R(n137), .S(1'b1),
      .Q(deadCount_captured[7]) );
DFFSR deadCount_captured_reg_6_ ( .D(N352), .CLK(clk), .R(n137), .S(1'b1),
      .Q(deadCount_captured[6]) );
DFFSR deadCount_captured_reg_5_ ( .D(N351), .CLK(clk), .R(n137), .S(1'b1),
      .Q(deadCount_captured[5]) );
DFFSR deadCount_captured_reg_4_ ( .D(N350), .CLK(clk), .R(n137), .S(1'b1),
      .Q(deadCount_captured[4]) );
DFFSR deadCount_captured_reg_3_ ( .D(N349), .CLK(clk), .R(n138), .S(1'b1),
      .Q(deadCount_captured[3]) );
DFFSR deadCount_captured_reg_2_ ( .D(N348), .CLK(clk), .R(n138), .S(1'b1),
      .Q(deadCount_captured[2]) );
DFFSR deadCount_captured_reg_1_ ( .D(N347), .CLK(clk), .R(n138), .S(1'b1),
      .Q(deadCount_captured[1]) );
DFFSR deadCount_captured_reg_0_ ( .D(N346), .CLK(clk), .R(n138), .S(1'b1),
      .Q(deadCount_captured[0]) );
DFFSR count_captured_reg_9_ ( .D(N345), .CLK(clk), .R(n138), .S(1'b1), .Q(
      count_captured[9]) );
DFFSR count_captured_reg_8_ ( .D(N344), .CLK(clk), .R(n138), .S(1'b1), .Q(
      count_captured[8]) );
DFFSR count_captured_reg_7_ ( .D(N343), .CLK(clk), .R(n138), .S(1'b1), .Q(
      count_captured[7]) );
DFFSR count_captured_reg_6_ ( .D(N342), .CLK(clk), .R(n138), .S(1'b1), .Q(
      count_captured[6]) );
DFFSR count_captured_reg_5_ ( .D(N341), .CLK(clk), .R(n138), .S(1'b1), .Q(
      count_captured[5]) );
DFFSR count_captured_reg_4_ ( .D(N340), .CLK(clk), .R(n138), .S(1'b1), .Q(
      count_captured[4]) );
DFFSR count_captured_reg_3_ ( .D(N339), .CLK(clk), .R(n138), .S(1'b1), .Q(
      count_captured[3]) );
```

```
DFFSR count_captured_reg_2_ ( .D(N338), .CLK(clk), .R(n138), .S(1'b1), .Q(
      count_captured[2]) );
DFFSR count_captured_reg_1_ ( .D(N337), .CLK(clk), .R(n139), .S(1'b1), .Q(
      count_captured[1]) );
DFFSR count_captured_reg_0_ ( .D(N336), .CLK(clk), .R(n129), .S(1'b1), .Q(
      count_captured[0]) );
AND2X2 U129 ( .A(totalCounts_bcd[8]), .B(n167), .Y(n245) );
AND2X2 U130 ( .A(n251), .B(totalCounts_bcd[4]), .Y(n250) );
AND2X2 U131 ( .A(n283), .B(counts_bcd[4]), .Y(n282) );
AND2X2 U132 ( .A(n1), .B(n285), .Y(n283) );
AND2X2 U133 ( .A(counts_bcd[8]), .B(n200), .Y(n274) );
AND2X2 U134 ( .A(n1), .B(counts_bcd[0]), .Y(n293) );
AND2X2 U135 ( .A(deadCounts_bcd[4]), .B(n315), .Y(n314) );
AND2X2 U136 ( .A(n189), .B(n419), .Y(n326) );
AND2X2 U138 ( .A(N76), .B(n140), .Y(N137) );
AND2X2 U139 ( .A(N75), .B(n140), .Y(N136) );
AND2X2 U140 ( .A(N74), .B(n140), .Y(N135) );
AND2X2 U141 ( .A(N73), .B(n140), .Y(N134) );
AND2X2 U142 ( .A(N72), .B(n140), .Y(N133) );
AND2X2 U143 ( .A(N71), .B(n140), .Y(N132) );
AND2X2 U144 ( .A(N70), .B(n140), .Y(N131) );
AND2X2 U145 ( .A(N69), .B(n140), .Y(N130) );
AND2X2 U146 ( .A(N68), .B(n140), .Y(N129) );
AND2X2 U147 ( .A(N67), .B(n140), .Y(N128) );
AND2X2 U148 ( .A(N66), .B(n140), .Y(N127) );
AND2X2 U149 ( .A(N65), .B(n140), .Y(N126) );
AND2X2 U150 ( .A(N64), .B(n140), .Y(N125) );
AND2X2 U151 ( .A(N63), .B(n140), .Y(N124) );
AND2X2 U152 ( .A(N62), .B(n140), .Y(N123) );
AND2X2 U153 ( .A(N61), .B(n140), .Y(N122) );
AND2X2 U154 ( .A(N60), .B(n140), .Y(N121) );
AND2X2 U155 ( .A(N59), .B(n140), .Y(N120) );
AND2X2 U156 ( .A(N58), .B(n140), .Y(N119) );
AND2X2 U157 ( .A(N57), .B(n140), .Y(N118) );
AND2X2 U158 ( .A(N56), .B(n140), .Y(N117) );
AND2X2 U159 ( .A(N55), .B(n140), .Y(N116) );
AND2X2 U160 ( .A(N54), .B(n140), .Y(N115) );
AND2X2 U161 ( .A(N53), .B(n140), .Y(N114) );
```

```
AND2X2 U162 ( .A(N52), .B(n140), .Y(N113) );
AND2X2 U163 ( .A(N51), .B(n140), .Y(N112) );
AND2X2 U164 ( .A(N50), .B(n140), .Y(N111) );
AND2X2 U165 ( .A(N49), .B(n140), .Y(N110) );
AND2X2 U166 ( .A(N48), .B(n140), .Y(N109) );
AND2X2 U167 ( .A(N47), .B(n140), .Y(N108) );
AND2X2 U168 ( .A(start_latch_counter), .B(N45), .Y(N107) );
INVX2 U266 ( .A(sendSerialData_n), .Y(n226) );
NOR2X1 U267 ( .A(systemState[0]), .B(n192), .Y(pausedIndicator) );
AOI22X1 U268 ( .A(N158), .B(n4), .C(totalCounts[9]), .D(n127), .Y(n227) );
AOI22X1 U269 ( .A(N157), .B(n4), .C(totalCounts[8]), .D(n127), .Y(n229) );
AOI22X1 U270 ( .A(N156), .B(n4), .C(totalCounts[7]), .D(n127), .Y(n230) );
AOI22X1 U271 ( .A(N155), .B(n4), .C(totalCounts[6]), .D(n127), .Y(n231) );
AOI22X1 U272 ( .A(N154), .B(n4), .C(totalCounts[5]), .D(n127), .Y(n232) );
AOI22X1 U273 ( .A(N153), .B(n4), .C(totalCounts[4]), .D(n127), .Y(n233) );
AOI22X1 U274 ( .A(N152), .B(n4), .C(totalCounts[3]), .D(n127), .Y(n234) );
AOI22X1 U275 ( .A(N151), .B(n4), .C(totalCounts[2]), .D(n127), .Y(n235) );
AOI22X1 U276 ( .A(N150), .B(n4), .C(totalCounts[1]), .D(n127), .Y(n236) );
AOI22X1 U277 ( .A(N149), .B(n4), .C(totalCounts[0]), .D(n127), .Y(n237) );
OAI21X1 U278 ( .A(n238), .B(n451), .C(n239), .Y(n368) );
NAND3X1 U279 ( .A(totalCounts_bcd[9]), .B(totalCounts_bcd[10]), .C(n240),
      .Y(n239) );
NOR2X1 U280 ( .A(totalCounts_bcd[11]), .B(n241), .Y(n240) );
AOI21X1 U281 ( .A(n242), .B(n450), .C(n243), .Y(n238) );
OAI21X1 U282 ( .A(n166), .B(n450), .C(n244), .Y(n369) );
NAND3X1 U283 ( .A(n165), .B(n450), .C(totalCounts_bcd[9]), .Y(n244) );
OAI21X1 U284 ( .A(totalCounts_bcd[9]), .B(n448), .C(n245), .Y(n243) );
OAI22X1 U285 ( .A(n245), .B(n449), .C(totalCounts_bcd[9]), .D(n241), .Y(n370) );
NAND3X1 U286 ( .A(n167), .B(n242), .C(totalCounts_bcd[8]), .Y(n241) );
NAND3X1 U287 ( .A(n450), .B(n449), .C(totalCounts_bcd[11]), .Y(n242) );
XOR2X1 U288 ( .A(totalCounts_bcd[8]), .B(n167), .Y(n371) );
NAND3X1 U289 ( .A(n440), .B(n4), .C(n443), .Y(n246) );
OAI21X1 U290 ( .A(n180), .B(n444), .C(n247), .Y(n372) );
NAND3X1 U291 ( .A(n181), .B(n444), .C(totalCounts_bcd[5]), .Y(n247) );
OAI22X1 U292 ( .A(n250), .B(n446), .C(totalCounts_bcd[5]), .D(n248), .Y(n373) );
XOR2X1 U293 ( .A(totalCounts_bcd[4]), .B(n251), .Y(n374) );
OAI21X1 U294 ( .A(n252), .B(n445), .C(n253), .Y(n375) );
NAND3X1 U295 ( .A(totalCounts_bcd[6]), .B(totalCounts_bcd[5]), .C(n254), .Y(
```

```
      n253) );
  NOR2X1 U296 ( .A(totalCounts_bcd[7]), .B(n248), .Y(n254) );
  NAND3X1 U297 ( .A(totalCounts_bcd[4]), .B(n255), .C(n251), .Y(n248) );
  AOI21X1 U298 ( .A(n255), .B(n444), .C(n249), .Y(n252) );
  OAI21X1 U299 ( .A(totalCounts_bcd[5]), .B(n443), .C(n250), .Y(n249) );
  NOR2X1 U300 ( .A(n127), .B(n256), .Y(n251) );
  NAND3X1 U301 ( .A(totalCounts_bcd[7]), .B(totalCounts_bcd[4]), .C(n257), .Y(
      n255) );
  NOR2X1 U302 ( .A(totalCounts_bcd[6]), .B(totalCounts_bcd[5]), .Y(n257) );
  OAI21X1 U303 ( .A(n258), .B(n442), .C(n259), .Y(n376) );
  NAND3X1 U304 ( .A(n179), .B(n442), .C(totalCounts_bcd[2]), .Y(n259) );
  AOI21X1 U305 ( .A(n261), .B(n447), .C(n262), .Y(n258) );
  OAI22X1 U306 ( .A(n182), .B(n447), .C(totalCounts_bcd[2]), .D(n260), .Y(n377) );
  NAND3X1 U307 ( .A(n256), .B(totalCounts_bcd[0]), .C(totalCounts_bcd[1]), .Y(
      n260) );
  OAI21X1 U308 ( .A(totalCounts_bcd[1]), .B(n440), .C(n263), .Y(n262) );
  OAI21X1 U309 ( .A(n263), .B(n441), .C(n264), .Y(n378) );
  NAND3X1 U310 ( .A(totalCounts_bcd[0]), .B(n441), .C(n256), .Y(n264) );
  NOR2X1 U311 ( .A(n439), .B(n127), .Y(n263) );
  AOI22X1 U312 ( .A(n439), .B(n256), .C(n127), .D(totalCounts_bcd[0]), .Y(n265) );
  NOR2X1 U313 ( .A(n127), .B(n440), .Y(n256) );
  NAND3X1 U314 ( .A(totalCounts_bcd[3]), .B(totalCounts_bcd[0]), .C(n266), .Y(
      n261) );
  NOR2X1 U315 ( .A(totalCounts_bcd[2]), .B(totalCounts_bcd[1]), .Y(n266) );
  OAI21X1 U316 ( .A(n267), .B(n438), .C(n268), .Y(n379) );
  NAND3X1 U317 ( .A(counts_bcd[9]), .B(counts_bcd[10]), .C(n269), .Y(n268) );
  NOR2X1 U318 ( .A(counts_bcd[11]), .B(n270), .Y(n269) );
  AOI21X1 U319 ( .A(n271), .B(n437), .C(n272), .Y(n267) );
  OAI21X1 U320 ( .A(n199), .B(n437), .C(n273), .Y(n380) );
  NAND3X1 U321 ( .A(n198), .B(n437), .C(counts_bcd[9]), .Y(n273) );
  OAI21X1 U322 ( .A(counts_bcd[9]), .B(n435), .C(n274), .Y(n272) );
  XOR2X1 U323 ( .A(counts_bcd[8]), .B(n200), .Y(n381) );
  OAI21X1 U324 ( .A(n275), .B(n432), .C(n276), .Y(n382) );
  NAND3X1 U325 ( .A(counts_bcd[6]), .B(counts_bcd[5]), .C(n277), .Y(n276) );
  NOR2X1 U326 ( .A(counts_bcd[7]), .B(n278), .Y(n277) );
  AOI21X1 U327 ( .A(n279), .B(n433), .C(n280), .Y(n275) );
  OAI21X1 U328 ( .A(n194), .B(n433), .C(n281), .Y(n383) );
  NAND3X1 U329 ( .A(n195), .B(n433), .C(counts_bcd[5]), .Y(n281) );
```

```
OAI21X1 U330 ( .A(counts_bcd[5]), .B(n430), .C(n282), .Y(n280) );
OAI22X1 U331 ( .A(n282), .B(n431), .C(counts_bcd[5]), .D(n278), .Y(n384) );
NAND3X1 U332 ( .A(counts_bcd[4]), .B(n279), .C(n283), .Y(n278) );
XOR2X1 U333 ( .A(counts_bcd[4]), .B(n283), .Y(n385) );
OAI22X1 U334 ( .A(n274), .B(n436), .C(counts_bcd[9]), .D(n270), .Y(n386) );
NAND3X1 U335 ( .A(n200), .B(n271), .C(counts_bcd[8]), .Y(n270) );
NAND3X1 U336 ( .A(n437), .B(n436), .C(counts_bcd[11]), .Y(n271) );
NAND3X1 U337 ( .A(n427), .B(n430), .C(n1), .Y(n286) );
NAND3X1 U338 ( .A(counts_bcd[7]), .B(counts_bcd[4]), .C(n287), .Y(n279) );
NOR2X1 U339 ( .A(counts_bcd[6]), .B(counts_bcd[5]), .Y(n287) );
OAI21X1 U340 ( .A(n288), .B(n429), .C(n289), .Y(n387) );
NAND3X1 U341 ( .A(n196), .B(n429), .C(counts_bcd[2]), .Y(n289) );
AOI21X1 U342 ( .A(n291), .B(n434), .C(n292), .Y(n288) );
OAI22X1 U343 ( .A(n193), .B(n434), .C(counts_bcd[2]), .D(n290), .Y(n388) );
NAND3X1 U344 ( .A(n197), .B(counts_bcd[0]), .C(counts_bcd[1]), .Y(n290) );
OAI21X1 U345 ( .A(counts_bcd[1]), .B(n427), .C(n293), .Y(n292) );
OAI21X1 U346 ( .A(n293), .B(n428), .C(n294), .Y(n389) );
NAND3X1 U347 ( .A(counts_bcd[0]), .B(n428), .C(n197), .Y(n294) );
OAI22X1 U348 ( .A(n1), .B(n426), .C(counts_bcd[0]), .D(n285), .Y(n390) );
NAND2X1 U349 ( .A(n1), .B(n291), .Y(n285) );
NAND3X1 U350 ( .A(counts_bcd[3]), .B(counts_bcd[0]), .C(n295), .Y(n291) );
NOR2X1 U351 ( .A(counts_bcd[2]), .B(counts_bcd[1]), .Y(n295) );
OAI21X1 U352 ( .A(n183), .B(n425), .C(n296), .Y(n391) );
NAND2X1 U353 ( .A(N272), .B(n183), .Y(n296) );
OAI21X1 U354 ( .A(n183), .B(n424), .C(n297), .Y(n392) );
NAND2X1 U355 ( .A(N271), .B(n183), .Y(n297) );
OAI21X1 U356 ( .A(n183), .B(n423), .C(n298), .Y(n393) );
NAND2X1 U357 ( .A(N270), .B(n183), .Y(n298) );
OAI21X1 U358 ( .A(n183), .B(n422), .C(n299), .Y(n394) );
NAND2X1 U359 ( .A(N269), .B(n183), .Y(n299) );
OAI21X1 U360 ( .A(n183), .B(n421), .C(n300), .Y(n395) );
NAND2X1 U361 ( .A(N268), .B(n183), .Y(n300) );
OAI21X1 U362 ( .A(n183), .B(n420), .C(n301), .Y(n396) );
NAND2X1 U363 ( .A(N267), .B(n183), .Y(n301) );
OAI21X1 U364 ( .A(n183), .B(n367), .C(n302), .Y(n397) );
NAND2X1 U365 ( .A(N266), .B(n183), .Y(n302) );
OAI21X1 U366 ( .A(n183), .B(n344), .C(n303), .Y(n398) );
NAND2X1 U367 ( .A(N265), .B(n183), .Y(n303) );
```

```
OAI21X1 U368 ( .A(n183), .B(n284), .C(n304), .Y(n399) );
NAND2X1 U369 ( .A(N264), .B(n183), .Y(n304) );
OAI21X1 U370 ( .A(n183), .B(n225), .C(n305), .Y(n400) );
NAND2X1 U371 ( .A(N263), .B(n183), .Y(n305) );
OAI21X1 U372 ( .A(n307), .B(n223), .C(n308), .Y(n401) );
NAND3X1 U373 ( .A(deadCounts_bcd[6]), .B(deadCounts_bcd[5]), .C(n309), .Y(
      n308) );
NOR2X1 U374 ( .A(deadCounts_bcd[7]), .B(n310), .Y(n309) );
AOI21X1 U375 ( .A(n311), .B(n222), .C(n312), .Y(n307) );
OAI21X1 U376 ( .A(n186), .B(n222), .C(n313), .Y(n402) );
NAND3X1 U377 ( .A(n187), .B(n222), .C(deadCounts_bcd[5]), .Y(n313) );
OAI21X1 U378 ( .A(deadCounts_bcd[5]), .B(n220), .C(n314), .Y(n312) );
OAI22X1 U379 ( .A(n314), .B(n221), .C(deadCounts_bcd[5]), .D(n310), .Y(n403)
       );
NAND3X1 U380 ( .A(n315), .B(n311), .C(deadCounts_bcd[4]), .Y(n310) );
NAND3X1 U381 ( .A(n221), .B(n222), .C(deadCounts_bcd[7]), .Y(n311) );
XOR2X1 U382 ( .A(deadCounts_bcd[4]), .B(n315), .Y(n404) );
NOR2X1 U383 ( .A(n306), .B(n316), .Y(n315) );
OAI21X1 U384 ( .A(n317), .B(n219), .C(n318), .Y(n405) );
NAND3X1 U385 ( .A(n185), .B(n219), .C(deadCounts_bcd[2]), .Y(n318) );
AOI21X1 U386 ( .A(n320), .B(n224), .C(n321), .Y(n317) );
OAI22X1 U387 ( .A(n188), .B(n224), .C(deadCounts_bcd[2]), .D(n319), .Y(n406)
       );
NAND3X1 U388 ( .A(n316), .B(deadCounts_bcd[0]), .C(deadCounts_bcd[1]), .Y(
      n319) );
OAI21X1 U389 ( .A(deadCounts_bcd[1]), .B(n217), .C(n322), .Y(n321) );
OAI21X1 U390 ( .A(n322), .B(n218), .C(n323), .Y(n407) );
NAND3X1 U391 ( .A(deadCounts_bcd[0]), .B(n218), .C(n316), .Y(n323) );
NOR2X1 U392 ( .A(n216), .B(n306), .Y(n322) );
AOI22X1 U393 ( .A(n216), .B(n316), .C(n306), .D(deadCounts_bcd[0]), .Y(n324)
       );
NOR2X1 U394 ( .A(n306), .B(n217), .Y(n316) );
NAND3X1 U395 ( .A(deadCounts_bcd[3]), .B(deadCounts_bcd[0]), .C(n325), .Y(
      n320) );
NOR2X1 U396 ( .A(deadCounts_bcd[2]), .B(deadCounts_bcd[1]), .Y(n325) );
NAND3X1 U397 ( .A(trigger_latched), .B(runStateIndicator), .C(n326), .Y(n306) );
OAI21X1 U398 ( .A(n1), .B(n215), .C(n327), .Y(n408) );
NAND2X1 U399 ( .A(N222), .B(n1), .Y(n327) );
```

```
OAI21X1 U400 ( .A(n1), .B(n214), .C(n328), .Y(n409) );
NAND2X1 U401 ( .A(N221), .B(n1), .Y(n328) );
OAI21X1 U402 ( .A(n1), .B(n213), .C(n329), .Y(n410) );
NAND2X1 U403 ( .A(N220), .B(n1), .Y(n329) );
OAI21X1 U404 ( .A(n1), .B(n212), .C(n330), .Y(n411) );
NAND2X1 U405 ( .A(N219), .B(n1), .Y(n330) );
OAI21X1 U406 ( .A(n1), .B(n211), .C(n331), .Y(n412) );
NAND2X1 U407 ( .A(N218), .B(n1), .Y(n331) );
OAI21X1 U408 ( .A(n1), .B(n210), .C(n332), .Y(n413) );
NAND2X1 U409 ( .A(N217), .B(n1), .Y(n332) );
OAI21X1 U410 ( .A(n1), .B(n209), .C(n333), .Y(n414) );
NAND2X1 U411 ( .A(N216), .B(n1), .Y(n333) );
OAI21X1 U412 ( .A(n1), .B(n208), .C(n334), .Y(n415) );
NAND2X1 U413 ( .A(N215), .B(n1), .Y(n334) );
OAI21X1 U414 ( .A(n1), .B(n207), .C(n335), .Y(n416) );
NAND2X1 U415 ( .A(N214), .B(n1), .Y(n335) );
OAI21X1 U416 ( .A(n1), .B(n206), .C(n336), .Y(n417) );
NAND2X1 U417 ( .A(N213), .B(n1), .Y(n336) );
NOR2X1 U418 ( .A(trigger_latched_dly), .B(n201), .Y(n337) );
OAI22X1 U419 ( .A(n191), .B(n202), .C(startCount_n), .D(n338), .Y(n418) );
AOI22X1 U420 ( .A(n338), .B(systemState[1]), .C(startCount_n), .D(n191), .Y(
      n339) );
NAND3X1 U421 ( .A(startCount_n_dly), .B(n340), .C(stopCount_n_dly), .Y(n338)
       );
XOR2X1 U422 ( .A(stopCount_n), .B(startCount_n), .Y(n340) );
OAI21X1 U423 ( .A(n341), .B(n203), .C(n342), .Y(N38) );
OAI21X1 U424 ( .A(n204), .B(n341), .C(n342), .Y(N37) );
OAI21X1 U425 ( .A(n341), .B(n205), .C(n342), .Y(N36) );
NAND2X1 U426 ( .A(n4), .B(n203), .Y(n342) );
NAND3X1 U427 ( .A(n419), .B(n189), .C(runStateIndicator), .Y(n228) );
NOR2X1 U428 ( .A(paddle_in_n[0]), .B(n343), .Y(n419) );
OR2X1 U429 ( .A(paddle_in_n[2]), .B(paddle_in_n[1]), .Y(n343) );
OR2X1 U430 ( .A(finishedCount), .B(n201), .Y(n341) );
NOR2X1 U431 ( .A(n202), .B(systemState[1]), .Y(runStateIndicator) );
OAI21X1 U432 ( .A(n425), .B(n141), .C(n345), .Y(N355) );
NAND2X1 U433 ( .A(deadCount_captured[9]), .B(n142), .Y(n345) );
OAI21X1 U434 ( .A(n424), .B(n141), .C(n346), .Y(N354) );
NAND2X1 U435 ( .A(deadCount_captured[8]), .B(n142), .Y(n346) );
```

```
OAI21X1 U436 ( .A(n423), .B(n141), .C(n347), .Y(N353) );
NAND2X1 U437 ( .A(deadCount_captured[7]), .B(n142), .Y(n347) );
OAI21X1 U438 ( .A(n422), .B(n141), .C(n348), .Y(N352) );
NAND2X1 U439 ( .A(deadCount_captured[6]), .B(n142), .Y(n348) );
OAI21X1 U440 ( .A(n421), .B(n142), .C(n349), .Y(N351) );
NAND2X1 U441 ( .A(deadCount_captured[5]), .B(n142), .Y(n349) );
OAI21X1 U442 ( .A(n420), .B(n142), .C(n350), .Y(N350) );
NAND2X1 U443 ( .A(deadCount_captured[4]), .B(n142), .Y(n350) );
OAI21X1 U444 ( .A(n367), .B(n142), .C(n351), .Y(N349) );
NAND2X1 U445 ( .A(deadCount_captured[3]), .B(n141), .Y(n351) );
OAI21X1 U446 ( .A(n344), .B(n142), .C(n352), .Y(N348) );
NAND2X1 U447 ( .A(deadCount_captured[2]), .B(n142), .Y(n352) );
OAI21X1 U448 ( .A(n284), .B(n142), .C(n353), .Y(N347) );
NAND2X1 U449 ( .A(deadCount_captured[1]), .B(n141), .Y(n353) );
OAI21X1 U450 ( .A(n225), .B(n141), .C(n354), .Y(N346) );
NAND2X1 U451 ( .A(deadCount_captured[0]), .B(n141), .Y(n354) );
OAI21X1 U452 ( .A(n215), .B(n142), .C(n355), .Y(N345) );
NAND2X1 U453 ( .A(count_captured[9]), .B(n142), .Y(n355) );
OAI21X1 U454 ( .A(n214), .B(n142), .C(n356), .Y(N344) );
NAND2X1 U455 ( .A(count_captured[8]), .B(n141), .Y(n356) );
OAI21X1 U456 ( .A(n213), .B(n141), .C(n357), .Y(N343) );
NAND2X1 U457 ( .A(count_captured[7]), .B(n142), .Y(n357) );
OAI21X1 U458 ( .A(n212), .B(n142), .C(n358), .Y(N342) );
NAND2X1 U459 ( .A(count_captured[6]), .B(n141), .Y(n358) );
OAI21X1 U460 ( .A(n211), .B(n141), .C(n359), .Y(N341) );
NAND2X1 U461 ( .A(count_captured[5]), .B(n141), .Y(n359) );
OAI21X1 U462 ( .A(n210), .B(n141), .C(n360), .Y(N340) );
NAND2X1 U463 ( .A(count_captured[4]), .B(n142), .Y(n360) );
OAI21X1 U464 ( .A(n209), .B(n141), .C(n361), .Y(N339) );
NAND2X1 U465 ( .A(count_captured[3]), .B(n141), .Y(n361) );
OAI21X1 U466 ( .A(n208), .B(n141), .C(n362), .Y(N338) );
NAND2X1 U467 ( .A(count_captured[2]), .B(n142), .Y(n362) );
OAI21X1 U468 ( .A(n207), .B(n141), .C(n363), .Y(N337) );
NAND2X1 U469 ( .A(count_captured[1]), .B(n142), .Y(n363) );
OAI21X1 U470 ( .A(n206), .B(n141), .C(n364), .Y(N336) );
NAND2X1 U471 ( .A(count_captured[0]), .B(n142), .Y(n364) );
NOR2X1 U473 ( .A(serialDeadCount_trx_busy), .B(serialCount_trx_busy), .Y(
      n366) );
```

```
NOR2X1 U474 ( .A(sendSerialData_n), .B(sendSerialData_dly), .Y(n365) );
BCD_to_7seg_1 totalCountsConvert ( .bcd(totalCounts_bcd), .segs(
      display_totalCounts_out) );
BCD_to_7seg_0 countsConvert ( .bcd(counts_bcd), .segs(display_counts_out) );
BCD_to_7seg_2 deadCountsConvert ( .bcd(deadCounts_bcd), .segs(
      display_deadCounts_out) );
serialOut_1 serialCountOut ( .clk(clk), .rst(rst), .initiate(n226),
      .data_in(count_captured), .data_out(serial_count_out), .trx_busy(
      serialCount_trx_busy) );
serialOut_0 serialDeadCountOut ( .clk(clk), .rst(rst), .initiate(n226),
      .data_in(deadCount_captured), .data_out(serial_deadCount_out),
      .trx_busy(serialDeadCount_trx_busy) );
daq_DW01_inc_0 add_237 ( .A(deadCounts), .SUM({N272, N271, N270, N269, N268,
      N267, N266, N265, N264, N263}) );
daq_DW01_inc_1 add_215 ( .A(counts), .SUM({N222, N221, N220, N219, N218,
      N217, N216, N215, N214, N213}) );
daq_DW01_inc_2 add_193 ( .A(totalCounts), .SUM({N158, N157, N156, N155, N154,
      N153, N152, N151, N150, N149}) );
daq_DW01_inc_3 add_99 ( .A(latch_counter), .SUM({N76, N75, N74, N73, N72,
      N71, N70, N69, N68, N67, N66, N65, N64, N63, N62, N61, N60, N59, N58,
      N57, N56, N55, N54, N53, N52, N51, N50, N49, N48, N47}) );
AND2X2 U16 ( .A(n337), .B(trigger_latched), .Y(n1) );
INVX2 U17 ( .A(n2), .Y(n140) );
OR2X1 U127 ( .A(n205), .B(N45), .Y(n2) );
INVX2 U137 ( .A(n4), .Y(n127) );
INVX2 U169 ( .A(n228), .Y(n4) );
INVX2 U170 ( .A(n3), .Y(n142) );
INVX2 U171 ( .A(n3), .Y(n141) );
INVX2 U172 ( .A(n306), .Y(n183) );
BUFX2 U173 ( .A(n452), .Y(n138) );
BUFX2 U174 ( .A(n452), .Y(n137) );
BUFX2 U175 ( .A(n452), .Y(n136) );
BUFX2 U176 ( .A(n452), .Y(n135) );
BUFX2 U177 ( .A(n452), .Y(n133) );
BUFX2 U178 ( .A(n452), .Y(n132) );
BUFX2 U179 ( .A(n452), .Y(n131) );
BUFX2 U180 ( .A(n452), .Y(n130) );
BUFX2 U181 ( .A(n452), .Y(n129) );
```

```
BUFX2 U182 ( .A(n452), .Y(n134) );
BUFX2 U183 ( .A(n452), .Y(n139) );
AND2X2 U184 ( .A(n365), .B(n366), .Y(n3) );
INVX2 U185 ( .A(rst), .Y(n452) );
OR2X1 U186 ( .A(latch_counter[9]), .B(latch_counter[8]), .Y(n143) );
OR2X1 U187 ( .A(latch_counter[12]), .B(n143), .Y(n150) );
AND2X1 U188 ( .A(latch_counter[1]), .B(latch_counter[0]), .Y(n144) );
NAND3X1 U189 ( .A(latch_counter[3]), .B(latch_counter[2]), .C(n144), .Y(n148) );
AND2X1 U190 ( .A(latch_counter[5]), .B(latch_counter[4]), .Y(n145) );
NAND3X1 U191 ( .A(latch_counter[7]), .B(latch_counter[6]), .C(n145), .Y(n147) );
NOR2X1 U192 ( .A(latch_counter[11]), .B(latch_counter[10]), .Y(n146) );
OAI21X1 U193 ( .A(n148), .B(n147), .C(n146), .Y(n149) );
OAI21X1 U194 ( .A(n150), .B(n149), .C(latch_counter[13]), .Y(n151) );
NOR2X1 U195 ( .A(n163), .B(n151), .Y(n152) );
NAND3X1 U196 ( .A(latch_counter[16]), .B(latch_counter[15]), .C(n152), .Y(
      n153) );
OAI21X1 U197 ( .A(latch_counter[17]), .B(n164), .C(latch_counter[18]), .Y(
      n154) );
NAND2X1 U198 ( .A(n154), .B(n162), .Y(n155) );
NAND3X1 U199 ( .A(latch_counter[20]), .B(n155), .C(latch_counter[21]), .Y(
      n157) );
NAND3X1 U200 ( .A(latch_counter[23]), .B(latch_counter[22]), .C(
      latch_counter[24]), .Y(n156) );
OAI21X1 U201 ( .A(n157), .B(n156), .C(n161), .Y(n158) );
AOI21X1 U202 ( .A(latch_counter[26]), .B(n158), .C(latch_counter[27]), .Y(
      n160) );
NOR2X1 U203 ( .A(latch_counter[29]), .B(latch_counter[28]), .Y(n159) );
NAND2X1 U204 ( .A(n160), .B(n159), .Y(N45) );
INVX2 U205 ( .A(latch_counter[25]), .Y(n161) );
INVX2 U206 ( .A(latch_counter[19]), .Y(n162) );
INVX2 U207 ( .A(latch_counter[14]), .Y(n163) );
INVX2 U208 ( .A(n153), .Y(n164) );
INVX2 U209 ( .A(n241), .Y(n165) );
INVX2 U210 ( .A(n243), .Y(n166) );
INVX2 U211 ( .A(n246), .Y(n167) );
INVX2 U212 ( .A(n237), .Y(n168) );
INVX2 U213 ( .A(n236), .Y(n169) );
INVX2 U214 ( .A(n235), .Y(n170) );
```

```
INVX2 U215 ( .A(n234), .Y(n171) );
INVX2 U216 ( .A(n233), .Y(n172) );
INVX2 U217 ( .A(n232), .Y(n173) );
INVX2 U218 ( .A(n231), .Y(n174) );
INVX2 U219 ( .A(n230), .Y(n175) );
INVX2 U220 ( .A(n229), .Y(n176) );
INVX2 U221 ( .A(n227), .Y(n177) );
INVX2 U222 ( .A(n265), .Y(n178) );
INVX2 U223 ( .A(n260), .Y(n179) );
INVX2 U224 ( .A(n249), .Y(n180) );
INVX2 U225 ( .A(n248), .Y(n181) );
INVX2 U226 ( .A(n262), .Y(n182) );
INVX2 U227 ( .A(n324), .Y(n184) );
INVX2 U228 ( .A(n319), .Y(n185) );
INVX2 U229 ( .A(n312), .Y(n186) );
INVX2 U230 ( .A(n310), .Y(n187) );
INVX2 U231 ( .A(n321), .Y(n188) );
INVX2 U232 ( .A(trigger_dly), .Y(n189) );
INVX2 U233 ( .A(n339), .Y(n190) );
INVX2 U234 ( .A(n338), .Y(n191) );
INVX2 U235 ( .A(systemState[1]), .Y(n192) );
INVX2 U236 ( .A(n292), .Y(n193) );
INVX2 U237 ( .A(n280), .Y(n194) );
INVX2 U238 ( .A(n278), .Y(n195) );
INVX2 U239 ( .A(n290), .Y(n196) );
INVX2 U240 ( .A(n285), .Y(n197) );
INVX2 U241 ( .A(n270), .Y(n198) );
INVX2 U242 ( .A(n272), .Y(n199) );
INVX2 U243 ( .A(n286), .Y(n200) );
INVX2 U244 ( .A(runStateIndicator), .Y(n201) );
INVX2 U245 ( .A(systemState[0]), .Y(n202) );
INVX2 U246 ( .A(busy), .Y(n203) );
INVX2 U247 ( .A(trigger_latched), .Y(n204) );
INVX2 U248 ( .A(start_latch_counter), .Y(n205) );
INVX2 U249 ( .A(counts[0]), .Y(n206) );
INVX2 U250 ( .A(counts[1]), .Y(n207) );
INVX2 U251 ( .A(counts[2]), .Y(n208) );
INVX2 U252 ( .A(counts[3]), .Y(n209) );
```

```
INVX2 U253 ( .A(counts[4]), .Y(n210) );
INVX2 U254 ( .A(counts[5]), .Y(n211) );
INVX2 U255 ( .A(counts[6]), .Y(n212) );
INVX2 U256 ( .A(counts[7]), .Y(n213) );
INVX2 U257 ( .A(counts[8]), .Y(n214) );
INVX2 U258 ( .A(counts[9]), .Y(n215) );
INVX2 U259 ( .A(deadCounts_bcd[0]), .Y(n216) );
INVX2 U260 ( .A(n320), .Y(n217) );
INVX2 U261 ( .A(deadCounts_bcd[1]), .Y(n218) );
INVX2 U262 ( .A(deadCounts_bcd[3]), .Y(n219) );
INVX2 U263 ( .A(n311), .Y(n220) );
INVX2 U264 ( .A(deadCounts_bcd[5]), .Y(n221) );
INVX2 U265 ( .A(deadCounts_bcd[6]), .Y(n222) );
INVX2 U472 ( .A(deadCounts_bcd[7]), .Y(n223) );
INVX2 U475 ( .A(deadCounts_bcd[2]), .Y(n224) );
INVX2 U476 ( .A(deadCounts[0]), .Y(n225) );
INVX2 U477 ( .A(deadCounts[1]), .Y(n284) );
INVX2 U478 ( .A(deadCounts[2]), .Y(n344) );
INVX2 U479 ( .A(deadCounts[3]), .Y(n367) );
INVX2 U480 ( .A(deadCounts[4]), .Y(n420) );
INVX2 U481 ( .A(deadCounts[5]), .Y(n421) );
INVX2 U482 ( .A(deadCounts[6]), .Y(n422) );
INVX2 U483 ( .A(deadCounts[7]), .Y(n423) );
INVX2 U484 ( .A(deadCounts[8]), .Y(n424) );
INVX2 U485 ( .A(deadCounts[9]), .Y(n425) );
INVX2 U486 ( .A(counts_bcd[0]), .Y(n426) );
INVX2 U487 ( .A(n291), .Y(n427) );
INVX2 U488 ( .A(counts_bcd[1]), .Y(n428) );
INVX2 U489 ( .A(counts_bcd[3]), .Y(n429) );
INVX2 U490 ( .A(n279), .Y(n430) );
INVX2 U491 ( .A(counts_bcd[5]), .Y(n431) );
INVX2 U492 ( .A(counts_bcd[7]), .Y(n432) );
INVX2 U493 ( .A(counts_bcd[6]), .Y(n433) );
INVX2 U494 ( .A(counts_bcd[2]), .Y(n434) );
INVX2 U495 ( .A(n271), .Y(n435) );
INVX2 U496 ( .A(counts_bcd[9]), .Y(n436) );
INVX2 U497 ( .A(counts_bcd[10]), .Y(n437) );
INVX2 U498 ( .A(counts_bcd[11]), .Y(n438) );
```

```
  INVX2 U499 ( .A(totalCounts_bcd[0]), .Y(n439) );
  INVX2 U500 ( .A(n261), .Y(n440) );
  INVX2 U501 ( .A(totalCounts_bcd[1]), .Y(n441) );
  INVX2 U502 ( .A(totalCounts_bcd[3]), .Y(n442) );
  INVX2 U503 ( .A(n255), .Y(n443) );
  INVX2 U504 ( .A(totalCounts_bcd[6]), .Y(n444) );
  INVX2 U505 ( .A(totalCounts_bcd[7]), .Y(n445) );
  INVX2 U506 ( .A(totalCounts_bcd[5]), .Y(n446) );
  INVX2 U507 ( .A(totalCounts_bcd[2]), .Y(n447) );
  INVX2 U508 ( .A(n242), .Y(n448) );
  INVX2 U509 ( .A(totalCounts_bcd[9]), .Y(n449) );
  INVX2 U510 ( .A(totalCounts_bcd[10]), .Y(n450) );
  INVX2 U511 ( .A(totalCounts_bcd[11]), .Y(n451) );
endmodule


module daq_top ( paddle_in_n, startCount_n, stopCount_n, rst_in_n, clk,
        sendSerialData_n, serialCount_trx_busy, serialDeadCount_trx_busy,
        runStateIndicator, pausedIndicator, display_totalCounts_out,
        display_counts_out, display_deadCounts_out, serial_count_out,
        serial_deadCount_out, counts, deadCounts, totalCounts, busy );
  input [2:0] paddle_in_n;
  output [20:0] display_totalCounts_out;
  output [20:0] display_counts_out;
  output [13:0] display_deadCounts_out;
  output [9:0] counts;
  output [9:0] deadCounts;
  output [9:0] totalCounts;
  input startCount_n, stopCount_n, rst_in_n, clk, sendSerialData_n;
  output serialCount_trx_busy, serialDeadCount_trx_busy, runStateIndicator,
         pausedIndicator, serial_count_out, serial_deadCount_out, busy;
  wire   startCount_gen_n, stopCount_gen_n, n10, n11, n12, n13, n14;
  wire   [2:0] paddle_in_syncd_n;
  wire   [2:0] reset_gen_n;

  DFFSR reset_gen_n_reg_0_ ( .D(1'b0), .CLK(clk), .R(1'b1), .S(n13), .Q(
        reset_gen_n[0]) );
  DFFSR reset_gen_n_reg_1_ ( .D(reset_gen_n[0]), .CLK(clk), .R(1'b1), .S(n13),
```

```
      .Q(reset_gen_n[1]) );
  DFFSR reset_gen_n_reg_2_ ( .D(reset_gen_n[1]), .CLK(clk), .R(1'b1), .S(n13),
      .Q(reset_gen_n[2]) );
  DFFSR stopCount_gen_n_reg ( .D(stopCount_n), .CLK(clk), .R(1'b1), .S(n14),
      .Q(stopCount_gen_n) );
  DFFSR paddle_in_syncd_n_reg_2_ ( .D(paddle_in_n[2]), .CLK(clk), .R(1'b1),
      .S(n14), .Q(paddle_in_syncd_n[2]) );
  DFFSR paddle_in_syncd_n_reg_1_ ( .D(paddle_in_n[1]), .CLK(clk), .R(1'b1),
      .S(n14), .Q(paddle_in_syncd_n[1]) );
  DFFSR paddle_in_syncd_n_reg_0_ ( .D(paddle_in_n[0]), .CLK(clk), .R(1'b1),
      .S(n14), .Q(paddle_in_syncd_n[0]) );
  DFFSR startCount_gen_n_reg ( .D(startCount_n), .CLK(clk), .R(1'b1), .S(n14),
      .Q(startCount_gen_n) );
  daq U1 ( .paddle_in_n(paddle_in_syncd_n), .startCount_n(startCount_gen_n),
      .stopCount_n(stopCount_gen_n), .rst(reset_gen_n[2]), .clk(clk),
      .sendSerialData_n(n11), .serialCount_trx_busy(serialCount_trx_busy),
      .serialDeadCount_trx_busy(serialDeadCount_trx_busy),
      .runStateIndicator(runStateIndicator), .pausedIndicator(
      pausedIndicator), .display_totalCounts_out(display_totalCounts_out),
      .display_counts_out(display_counts_out), .display_deadCounts_out(
      display_deadCounts_out), .serial_count_out(serial_count_out),
      .serial_deadCount_out(serial_deadCount_out), .counts(counts),
      .deadCounts(deadCounts), .totalCounts(totalCounts), .busy(busy) );
  INVX2 U13 ( .A(sendSerialData_n), .Y(n10) );
  INVX2 U14 ( .A(n10), .Y(n11) );
  INVX2 U15 ( .A(rst_in_n), .Y(n12) );
  INVX2 U16 ( .A(n12), .Y(n13) );
  INVX2 U17 ( .A(reset_gen_n[2]), .Y(n14) );
endmodule
```

Appendix D

Appendix D will briefly outline motivations for this research project.

My research group is developing a cosmic particle telescope to observe muon decays from cosmic particle interactions in the atmosphere. The muons are traveling at relativistic speeds (near the speed of light); however they have an extremely short half-life. Nearly all of them should decay before they reach ground level. Yet, they can still be detected in substantial portions. The observance of these particles demonstrates relativity in action (time dilation and length contraction). We want to use this project to demonstrate those concepts to high school students as part of an outreach program. Figure D.1[9] shows the muon production in the atmosphere.

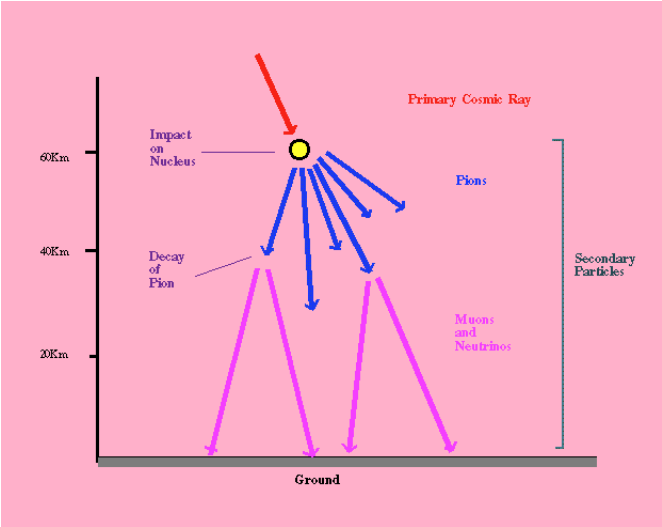

Figure D.1: Muon production from decay of pions.

My group's experiment will use three detectors. Each detector unit consists of a paddle shaped piece of scintillator plastic attached to a lightguide which is glued (optical glue) to a photomultiplier tube (PMT). The entire device is wrapped to prevent external light from entering. When a particle hits the paddle, energy is deposited and photons are generated in the plastic scintillator. These photons are guided to the PMT which outputs a negative analog pulse (dependent on number of photons generated which is dependent on energy deposited). Three such detector units are used in order to form a cone of incidence to ensure the particles are cosmic in origin. With only one detector, a trigger event could happen from a particle passing at angle. Using coincidence of three paddles means that the particle trajectory is coming from the atmosphere (altering spacing and sizing can narrow field of view).

[9] "What does the Spark Chamber Detect?" http://www.ep.ph.bham.ac.uk/general/outreach/SparkChamber/text7h.html.

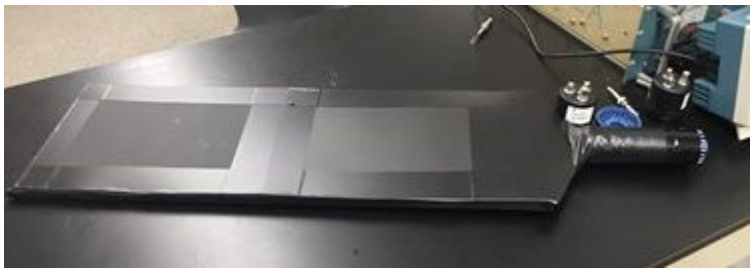

Figure D.2: One detector unit.

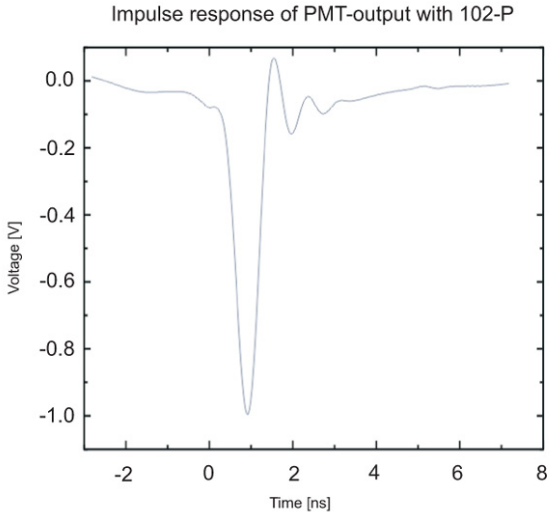

Figure D.3: Example of signal from PMT.

Before the PMT output signal can be sent to the FPGA, it must go through frontend electronics. This is another portion of the project I am working on. The simple version uses gain (negative gain to work with positive signals) and discrimination circuitry (op-amps) to increase the 30-100mv signal from the PMT to something closer to 3.3V. The discriminator compares the pulse to a voltage threshold to see if the signal is greater than background noise. These signals are then essentially digital signals and are the signals used in the FPGA design. This works for counting coincidence as is implemented in the project in this report.

The next step is to extract more meaningful information by capturing pulse shape. The minimum pulse widths from the PMTs are 10ns.[10] Not only do we want to capture pulse shape, but we want to integrate under the pulse (to extract information about charge and energy). This requires a high sample rate ADC and is currently introducing complications in the next steps of the design.

Alteratively, we may only want to capture signals of longer pulse widths (to simplify project). In this case, the FPGA's trigger logic could check for long enough pulse duration.

Another update using fast enough ADCs would be to perform all discrimination in digital domain instead of analog domain. This setup would alter the design configuration shown at the beginning of this report.

---

[10] "PAM 102," *PicoQuant*, https://www.picoquant.com/products/category/accessories/pam-102-pre-amplifier-module.